**TUGAS AKHIR – TE141599**

# *DYNAMIC ECONOMIC DISPATCH* (DED) DENGAN MEMPERHATIKAN *RAMP-RATE* MENGGUNAKAN METODE ITERASI LAMBDA BERBASIS DELPHI

**Hardi Rizkyanto**
**NRP 2211 100 143**

**Dosen Pembimbing**
**Prof. Ir. Ontoseno Penangsang, M. Sc, Ph. D.**
**Ir. Sjamsjul Anam, MT.**

**JURUSAN TEKNIK ELEKTRO**
**FakultasTeknologiIndustri**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2015**

Institut
Teknologi
Sepuluh Nopember

**TUGAS AKHIR – TE141599**

# *DYNAMIC ECONOMIC DISPATCH* (DED) DENGAN MEMPERHATIKAN *RAMP-RATE* MENGGUNAKAN METODE ITERASI LAMBDA BERBASIS DELPHI

**Hardi Rizkyanto**
**NRP 2211 100 143**

**Dosen Pembimbing**
**Prof. Ir. Ontoseno Penangsang, M. Sc, Ph. D.**
**Ir. Sjamsjul Anam, MT.**

**JURUSAN TEKNIK ELEKTRO**
**FakultasTeknologiIndustri**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2015**

Institut
Teknologi
Sepuluh Nopember

**FINAL PROJECT – TE 141599**

# DYNAMIC ECONOMIC DISPATCH (DED) CONSIDERING RAMP-RATE USING LAMBDA ITERATION BASED ON DELPHI

**Dwi Haryanto**
**NRP 2211100158**

**Advisor**
**Prof. Ir. Ontoseno Penangsang, M. Sc, Ph. D.**
**Ir. Sjamsjul Anam, MT.**

**ELECTRICAL ENGINEERING DEPARTEMENT**
**Faculty of Industrial Technology**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2015**

# *DYNAMIC ECONOMIC DISPATCH (DED)* DENGAN MEMPERHATIKAN *RAMP-RATE* MENGGUNAKAN METODE ITERASI LAMBDA BERBASIS DELPHI

**TUGAS AKHIR**

**Diajukan Guna Memenuhi Sebagian Persyaratan**
**Untuk Memperoleh Gelar Sarjana Teknik**
**Pada**
**Bidang Studi Teknik Sistem Tenaga**
**Jurusan Teknik Elektro**
**Institut Teknologi Sepuluh Nopember**

**Menyetujui:**

| Dosen Pembimbing I, | Dosen Pembimbing II, |
|---|---|
| **Prof. Ir. Ontoseno Penangsang, M.Sc., Ph.D.** | **Ir. Sjamsjul Anam, MT.** |
| **NIP : 194907151974121001** | **NIP :196307251990031002** |

**SURABAYA**
**JUNI, 2015**

# ABSTRAK

**_Dynamic Economic Dispatch_ (DED) dengan Memperhatikan _Ramp-rate_ Menggunakan Metode Iterasi Lambda Berbasis Delphi**

Hardi Rizkyanto
2211100143

Dosen pembimbing 1 :Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dosen Pembimbing 2 :Ir. Sjamsjul Anam, MT.

**Abstrak**

Pembagian pembebanan pembangkit yang bertujunan untuk meminimalkan biaya pembangkitan dikenal dengan istilah *Economic Dispatch* (ED). Perubahan beban yang dilihat dalam setiap interval waktu akan menghasilkan perhitungan *Dynamic Economic Dispatch* (DED). Pada Tugas Akhir ini memperhatikan *ramp-rate* pada perhitungan DED. *Ramp-rate* digunakan untuk merubah batasan yang akan digunakan dalam perhitungan DED pada interval waktu berikutnya, sehingga pembebanan yang diberikan kepada setiap pembangkit akan semakin selektif. Tugas Akhir menggunakan metoda iterasi lambda sebagai penyelesaian masalah optimalisasi biaya pada DED. Metoda iterasi lambda akan diterapkan pada aplikasi pemrograman Delphi untuk meng-*upgrade* aplikasi perhitungan PowerGen yang digunakan pada matakuliah Optimalisasi Sistem Tenaga Elekto ITS, dengan cara menambahkan fitur aplikasi perhitungan DED didalam *software*. Uji kebenaran akan didapatkan dengan melihat hasil pengoperasian aplikasi perhitungan yang telah dibuat tidak melanggar batasan *ramp-rate.*

**Kata kunci**: *Economic Dispatch, Dynamic Economic Dispatch, ramp-rate,* Delphi

# ABSTRACT

***Dynamic Economic Dispatch (DED) Considering Ramp-Rate Using Lambda Iteration Based on Delphi***

Hardi Rizkyanto
2211100143

Dosen pembimbing 1 :Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dosen Pembimbing 2 :Ir. Sjamsjul Anam, MT.

**Abstract**

Load-sharing power plants which aims to minimize the cost of the generation known as the Economic Dispatch (ED). Load changes seen in each time interval will result in the calculation of Dynamic Economic Dispatch (DED). In this final notice on the ramp-rate calculation DED. Ramp-rate is used to change the limit to be used in the calculation of DED in the next time interval, so that the load given to each generation will be more selective. The final task using lambda iteration method as cost optimization problem solving at DED. Lambda iteration method will be applied to the Delphi programming application to upgrade the PowerGen computing applications used on subjects Optimization of Power System Elekto ITS, by adding features DED calculation within software applications. Testing will be obtained by looking at the operation of the application calculations that have been made do not violate more than ramp-rate limit

**Index Term:** Economic Dispatch, Dynamic Economic Dispatch, ramp-rate, Delphi

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat, karunia, dan petunjuk yang telah dilimpahkan-Nya sehingga penulis mampu menyelesaikan Tugas Akhir dengan judul :

***Dynamic Economic Dispatch* (DED) dengan Memperhatikan *Ramp-rate* Menggunakan Metode Iterasi Lambda Berbasis Delphi**

Tugas Akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan S1 pada Bidang Studi Teknik Sistem Tenaga, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.

Pada kesempatan ini penulis hendak menyampaikan rasa terima kasih kepada pihak-pihak yang memberikan peranan penting dalam menyelesaikan Tugas Akhir ini, kepada:

1. Allah SWT atas limpahan Rahmat dan Petunjuk-Nya serta Nabi Muhammad SAW atas tuntunan jalan-Nya.
2. Bapak dan Ibu yang telah membesarkan, mendidik saya hingga dewasa kini.
3. Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D dan Dr. Rony Seto Wibowo, ST., MT. sebagai dosen pembimbing yang telah memberikan arahan dan perhatiannya dalam Tugas Akhir ini.
4. Seluruh dosen yang telah memberikan ilmunya selama kuliah, karyawan, dan keluarga besar Jurusan teknik Elektro ITS
5. Teman-teman Teknik Elektro ITS 2011 dan khususnya kepada teman-teman satu kelompok atas bantuan kalian selama masa pengerjaan Tugas Akhir ini
6. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung, yang tidak mungkin saya sebutkan satu per satu

Penulis menyadari bahwa Tugas Akhir ini belum sempurna, Oleh karena itu saran dan masukan sangat diharapkan untuk perbaikan dimasa yang akan datang.

Surabaya, Juli 2015

Penulis

# DAFTAR ISI

# DAFTAR TABEL

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Pengaturan penjadwalan pembangkit dibutuhkan oleh perusahaan untuk mendapatkan hasil perhitungan yang paling ekonomis sehingga perusahaan tidak mengalami kerugian. *Economic Dispatch* (ED) merupakan pembagian pembebanan pada unit-unit pembangkit yang ada dalam sistem secara optimal ekonomi pada harga beban sistem tertentu. Dengan penerapan *Economic Dispatch* maka akan didapatkan biaya pembangkitan yang minimum terhadap produksi daya listrik yang dibangkitkan unit-unit pembangkit pada suatu system kelistrikan. Pada Tugas Akhir ini memodifikasi persamaan yang ada pada ED berdasarkan perubahan beban di tiap waktunya. Persamaan tersebut dikenal sebagai *Dynamic Economic Dispatach* (DED) dengan memperhatikan *ramp-rate* yang sangat di pengaruhi oleh berbagai sistem yang terpasang di dalam pembangkit tersebut. Sebagai contoh, pada pembangkit dengan turbin uap berbahan bakar batu bara mempunyai level *ramp-rate* yang berbeda. Pengaruh dari *ramp-rate* yang merubah batasan yang ada pada persamaan untuk mendapatkan nilai optimal. Dalam referensi [1][2], menggunakan persamaan *ramp-rate* yang merupakan modifikasi pada batasan *maximum* dan *minimum* pada tiap unit berdasarakan perubahan beban, yang nantinya ketika dimasukkan persamaan akan menghasilkan DED yang berbeda ketika menggunkaan batasan sebelumnya akibat pengaruh perubahan batasan akibat *ramp-rate*.

Banyak metode yang digunakan dalam melakukan perhitungan ED. Salah satu metode yang digunakan pada Tugas Akhir ini adalah dengan menggunkan metode iterasi lambda [3]. Iterasi lambda sendiri memperhitungkan perubahan setiap lambda yang akan dimasukkan kedalam persamaan DED pada Tugas Akhir ini. Dikarenakan DED adalah jumlah perhitungan ED pada tiap waktu, maka pada Tugas Akhir ini kan didapatkan nilai lambda yang berfariasi tiap perubahan beban.

Simulasi yang digunakan pada Tugas Akhir ini diusulkan menggunakan Delphi. Delphi merupakan bahasa pemrograman berbasis Windows yang menyediakan fasilitas pembuatan aplikasi visual. Delphi memberikan kemudahan dalam menggunakan kode program, kompilasi yang cepat, penggunaan file unit ganda untuk pemrograman modular, pengembangan perangkat lunak, pola desain yang menarik serta diperkuat

dengan bahasa pemrograman yang terstruktur dalam bahasa pemrograman *Object Pascal*. Delphi memiliki tampilan khusus yang didukung suatu lingkup kerja komponen Delphi untuk membangun suatu aplikasi dengan menggunakan *Visual Component Library* (VCL). Sebagian besar pengembang Delphi menuliskan dan mengkompilasi kode program dalam *Integrated Development Environment* (IDE) [4]. Sehinga pada Tugas Akhir ini akan menghasilkan sebuah aplikasi perhitungan yang menggunakan bahasa yang dapat dipahami oleh pengguna. Aplikasi penyelesaian DED dengan memperhatikan *ramp-rate* yang lebih mudah untuk berinteraksi dengan pengguna.

## 1.2 Tujuan Penelitian

Tujuan yang ingin dicapai pada Tugas Akhir ini adalah sebagai berikut:

1. Membuat aplikasi perhitungan *Dynamic Economic Dispatch* (DED) dengan menggunkan metode iterasi lambda berbasis Delphi.
2. Melihat pengaruh *ramp-rate* pada perhitungan *Dynamic Economic Dispatch*.

## 1.3 Permasalahan

Dalam Tugas Akhir ini diharapkan permasalahan dapat terselesaikan, yaitu:

1. *Sofware* perhitungan DED yang telah dikembangkan dapat membantu pengguna dalam melakukan perhitungan DED dengan memperhatikan *ramp-rate* yang diseleasikan dengan metode iterasi lambda.
2. Memberikan pemahaman akan pengaruh *ramp-rate* dalam perhitungan DED

## 1.4 Batasan Masalah

Batasan masalah yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

1. Mengembangkan *sofware* perhitungan dikerjakan dengan menggunakan Delphi.
2. Metode yang digunakan untuk menyelasiakan permasalahan DED adalah metode itersi lambda
3. Perhitungan DED hanya memperhatikan pengaruh batasan *ramp-rate*.

4. Data rugi-rugi didapatkan dalam bentuk *matrix*.
5. Data rugi-rugi diperuntukkan untuk semua perhitungan DED di setiap periode.
6. Periode beban diasumsikan menggunakan interval waktu 1 jam
7. Data beban dalam aplikasi perhitungan DED tidak melebihi data total kemampuan pembangkit
8. Semua pembangkit diasumsikan selalu dalam keadaan menyala

## 1.5 Metode Penelitian

Pada Tugas Akhir ini dilakukan penelitian tentang *Dynamic Economi Dispatch* dengan memperhatikan batasan *ramp-rate* yang diselsaikan dengan menggunkan motode iterasi lambda. Perhitungan akan dilakukan dengan menggunkan aplikasi perhitungan berbasis Delphi. Tahap pengerjaan Tugas Akhir ini adalah:

1. Studi Literatur

   Studi literatur untuk mencari referensi bahan melalui buku, jurnal ilmiah (*paper*), dan *browsing* melalui internet yang berhubungan dengan judul Tugas Akhir ini. Referensi yang dicari mencangkup teori *Dynamic Economic Dispatch* yang melihat pengaruh dari *ramp-rate*, Metode iterasi lambda yang dapat diterap dalam penyelesaian *output* pembangkitan dalam persamaan DED.

2. Pengumpulan Data, Perhitungan dan

   Data yang bersangkutan dalam permasalahn DED seperti batasan *maximum* pembangkit, batasan *minimum* pembangkit, nilai *up-rate*, nilai *down-rate*, data beban tiap periode dalam interval waktu 1 jam, *fuelcost* dan nilai koefisien dari persamaan pembangkitan. Berdasarkan formulasi yang telah diketahui dilakukan perhitungan terlebih dahulu untuk menunjang pemodelan dan simulasi.

3. Perencanaan dalam Pembuatan Program

   Dari data perhitungan tersebut dibuatlah strukur perancangan logika berupa diagram alir untuk proses pembuatan program yang nantinya akan diimplemantasikan kedalam bahasa pemrograman Delphi.

4. Simulasi Analisa Data

   Setelah pembuatan program telah selesai dan menjadi sebuah aplikasi perhitungan, simulasi akan dicoba dengan menggunkan data inputan yang telah tersedia. Simulasi bertujuan untuk melihat aplikasi perhitungan DED yang dibuat telah sesuai dengan teori

perhitungan yang ada. Dari penjalanan simulasi akan didapatkan data yang akan dianalisa kebenarannya. Analisa bertujuan untuk memastikan hasil perhitungan program telah sesuai dengan metode yang digunakan. Data *output* akan kemudian dilihat apakah sesuai dengan batasan DED yang telah ditentukan.

5. Penulisan Buku

Hasil penelitian yang telah dilakukan akann dilaporkan dalam sebuah buku laporan Tugas Akhir. Isi dari laporan berdasarkan kesimpulan dari analisis yang telah didapat beserta tahapan yang ada di dalamnya.

## 1.6 Sistematika Penulisan

Sistematika penulisan dalam Tugas Akhir ini terdiri atas lima bab, dengan uraian sebagai berikut:

BAB 1 : Pendahuluan

Bagian ini membahas dasar-dasar penyusunan Tugas Akhir ini meliputi latar belakang, permasalahan yang diangkat, tujuan yang diharapkan, batasan masalah, metodologi pembuatan Tugas Akhir, sistematika dan relevansi penyusunan laporan Tugas Akhir ini.

BAB 2 : Tinjauan Pustaka

Bagian ini membahas teori-teori penunjang yang melandasi Tugas Akhir ini, formulasi perhitungan *Economic Dispatch* (ED), perkembangan ED yang disebut dengan *Dynamic Economic Dispatch* (DED), pengaruh *ramp-rate*, serta metode iterasi lambda sebagai metode yang digunakan dalam Tugas Akhir ini.

BAB 3 : Desain dan Simulasi

Bagian ini berisi proses desain, pemodelan serta simulasi yang dikerjakan agar didapatkan hasil aplikasi perhitungan *Dynamic Economic Dispatch* dengan memperhatikan *ramp-rate*, dimulai dengan cara melakukan perhitungan pada DED, kemudian dilanjutkan dengan proses penggunaan aplikasi perhitungan.

BAB 4 : Hasil Simulasi dan Analisis Data

Bagian ini membahas mengenai hasil simulasi yang didapatkan dari proses penyelesaian perhitungan yang dilakukan oleh aplikasi perhitungan yang telah dibuat,

serta analisa pengaruh *ramp-rate* pada perhitunga DED yang akan diuji dalam Tugas akhir ini.

BAB 5 : Penutup

Bagian ini membahas kesimpulan yang dapat diambil dari hasil penjalanan simulasi yang telah dilakukan analisa. Selain itu juga dilampirkan saran yang diharapkan mampu memberikan perbaikan serta penyempurnaan terkait keberlanjutan Tugas Akhir ini.

## 1.7 Relevansi

Hasil yang diperoleh dari Tugas Akhir ini diharapkan dapat memberikan kontribusi sebagai berikut

1. Dapat memberikan manfaat dalam kemudahan melakukan perhitungan *Dynamic Economic Dispatch* menggunakan iterasi lambda dengan memperhatikan *ramp-rate*.
2. Dapat menambah penguasaan ilmu dan teknologi di bidang optimalisasi pembangkitan sistem tenaga listrik.
3. Dapat menjadi reverensi Tugas Akhir untuk mahasiswa yang akan mengambil Tugas Akhir untuk mengembangkan aplikasi perhitungan DED dengan permasalahan yang berbeda.

*Halaman Ini Sengaja Dikosongkan*

# BAB 2
# DYNAMIC ECONOMIC DISPATCH

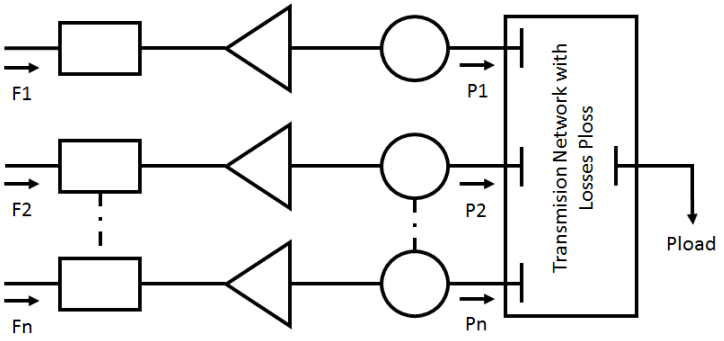
## 2.1 Sistem Tenaga Listrik

Sudah tidak dapat dipungkiri bahwa energi menempati peringkat yang sangat penting sebagai kebutuhan umat manusia. Sejak berabad-abad yang lalu setiap individu, kelompok maupun negara berjuang untuk memenuhi kebutuhannya akan energi. Hal tersebut mengakibatkan energi semakin langka dan harganya meningkat terus.

Salah satu bentuk energi yang sangat mudah dimanfaatkan adalah listrik. Pada era modern saat ini energi listrik ini menjadi factor penting bagi kehidupan manusia. Peralatan elektronik yang semakin berkembang menuntut manusia mengkonsumsi energi listrik untuk menunjang kebutuhan hidup sehari. Begitu juga dengn perusahaan modern saat ini yang menggunakan teknologi yang energi pengeraknya sendiri dari tenaga listrik.

Dilakukan berbagai kegitan untuk mengubah berbagai bentuk energi tersebut menjadi listrik dan kemudian sampai ke tangan konsumen. Proses tersebut dilakukan dalam sebuah kegiatan yang disebut dengan sistem tenaga listrik.

Sistem Tenaga Listrik adalah sistem penyediaan tenaga listrik yang terdiri dari beberapa pembangkit atau pusat listrik terhubung satu dengan lainnya oleh jaringan transmisi dengan pusat beban atau jaringan distribusi. Secara umum sistem tenaga listrik digambarkan dalam Gambar 2.1




**Gambar 2.1** N thermal unit mensuplai beban melalui jaringan transmisi

Pembangkit-pembangkit tenaga listrik dengan lokasi berjauhan satu sama lain terhubung pada sistem melalui sistem transmisi yang luas untuk mencatu tenaga listrik pada beban yang besar, disebut dengan sistem interkoneksi. Sistem interkoneksi menyebabkan:

1. Keandalan sistem yang semakin tinggi
2. Efisiensi pembangkittan tenaga listik dalam sistem meningkat
3. Mempermudah pejadwalan pembangkit

Kondidisi kesetimbangan antara pendapatan (penjualan) dan pengeluaran (pembiayaan) harus dijaga, agar dapat diperoleh keuntungan, sehingga kelangsungan unit usaha listrik dalam peruasahaan dapat terjaga.

Saat ini sistem tenaga mimiliki sistem intekoneksi *network* yang komplek [5]. Sitem tenaga dapat terdefinisikan bedasarkan empat bagian utama yang akan dijelaskan oleh subbab 2.1 berikut:

### 2.1.1 Generator

Salah satu bagian terpenting dari sistem tenaga listrik adalah pembangkit tenaga listrik. Stasiun pembangkit itu sendiri dapat berupa generator yang digerakkan dengan tenaga gas, tenaga air, tenaga disel dan lain sebagainya. Baiasanya jenis generator yang digunakan adalah generator 3 fasa atau generator sinkron. Sistem saat ini menggunakan generator AC dengan *rotating rectifier*, dikenal dengan sistem eksitasi *brushless*. Eksitasi generator bertujuan untuk menjaga tegangan pada generator dan mengontrol aliran daya reaktif. Akibat kekurangan komutator, generator AC dapat menghasilkan daya besar dengan tegangan tinggi sebesar 30 kV. Dalam *power plant*, ukuran daya yang dihasilkan berkisar antara 50 MW sampai 1500 MW.

Sumber tenaga mekanik biasa disebut dengan *prime mover* yang dapat berupa turbin *hydrolic* yang menggunakan tenaga potensial air terjun, turbin uap yang menggunakan sumber energi hasil pembakaran batu bara, dan lain- lain. Sumber-sumber energi alam dirubah oleh penggerak mula menjadi energi mekanis yang berupa kecepatan atau putaran, selanjutnya energi mekanis tersbut dirubah menjadi energi listrik oleh generator. Biasanya proses pembangkitan listrik oleh generator dihasilkan dengan menggunakan induksi elektromagnetik. Meskipun terdapat banyak kesamaan, generator berbeda dengan motor. Hal ini dikarenakan motor merupakan alat yang mengubah energi listrik menjadi energi mekanik. Sehingga dapat dikataka motor adalah beban yang justru membutuhkan energi listrik untuk mengerakkan rotor di dalamnya.

### 2.1.2 Transmisi dan Sub-Tranmisi

Transmisi listrik bertujuan untuk menyalurkan energi listrik dari unit pembangkit yang berasal dari berbagai tempat menuju sistem distribusi yang terhubung sebagai suplai kepada beban. Saluran transmisi juga menghubungkan peralatan selama sistem bekerja normal, maupun ketika terjadi gangguan.

Standart tegangan transmisi ditetapkan berdasarkan standart *American National Standart Institute* (ANSI). Tegangan transmisi bekerja pada jaringan lebih dari 60 kV distandarisasi menjadi 69 kv, 115 kv, 138 kv, 161 kv, 230 kv, 345 kv, 500 kv, dan 765 kv *line to line*. Tegangan tranmisi di atas 230 kv biasa disebut dengan tegangan ekstra tinggi.

Pada sistem transmisi yang terhubung dengan subtansi tegangan tinggi dilewatkan transformator *step-down* yang menuju subtansi distribusi. Beberapa industri besar memungkinkan untuk mendapatkan suplai langsung dari sistem sub-transmisi. Kapasitor bank dan reaktor bank biasanya terpasang pada subtansi untuk mempertahankan tegangan.

### 2.1.3 Distribusi

Sistem Distribusi merupakan bagian yang menghubungkan gardu induk distribusi pada konsumen sebagai masukan awal suplai tenaga terhadap beban. Besar tegangan saluran distribusi primer biasanya berkisar antaray 4 KV sampai dengan 34.5 KV dan mensuplai beban dalam area yang telah ditentukan.

Jaringan distribusi sekunder bertujuan untuk mengurangi tegangan yang diperuntukkan untuk penggunaan konsusmsi beban baik comersial maupun perumahan. Panjang kabel dipasang tidak melebihi 100 kaki yang terhubung dengan setiap konsumen. Distribusi sekunder kebanyakan melayani kosumen dengan level tegangan 240/120 V *single phase* tiga belitan, 280Y/12O V tiga *phase* empat belitan, atau 480Y/277 V tiga *phase* empat belitan.

Berdasarkan letak, sistem disribusi dibagi menjadi 2, yaitu:

1. *Overhead*, merupakan kabel atau kawat transmisi listrik disalurkan di udara atau di atas tanah. Memiliki kelebihan kemudahan dalam pemeliharaan dan kemudahan dalam perlusan wilayah, akan tetapi mudah mengalami gangguan.
2. *Underground*, merupakan kabel atau transimis listrik yang terletak di bawah tanah. Memiliki kelebihan tidak mudah mendapatkan gangguan, akan tetapi biaya pembangunan mahal

dan pemeliharaan lebih sulit karena letak yang berada di dalam tanah.

### 2.1.4 Beban Sistem

Daya beban real tercantum dalam satuan kilowatt (KW) atau megawatt (MW) dengan *magnitude* beban yang berfariasi setiap harinya, daya diharuskan mampu memenuhi kebutuhan beban. Masalah akan terjadi apabila daya yang dikirim lebih besar daripada kebutuhan beban, maka akan mengakibatkan kerugian pada perusahaan listrik akibat pemborosan energi. Sedangkan jika daya yang dikirim lebih kecil daripada permintaan beban, maka akan terjadi *over load* yang dapat mengakibatkan pemadaman.

Perkiraan beban merupakan masalah yang sangat menentukan bagi perusahaan listrik baik dalam segi manajerial maupun segi operasional. Oleh karenanya perlu mendapat perhatian khusus untuk dapat membuat perkiraan beban yang sebaik mungkin. Berdasarkan jangka waktunya perkiraan beban dapat dibedakan menjadi:

1. Perkiraan beban jangka panjang (*long term*), merupakan perkiraan beban listrik untuk jangka waktu di atas satu tahun.
2. Perkiraan beban jangka menengah (*medium term*), merupakan perkiraan beban lisrik untuk jangka waktu antara satu bulan sampai dengan satu tahun.
3. Perkiraan beban jangka pendek (*short term*), merupkan perkiraan beban listrik untuk jangka waktu beberapa jam dalam sehari sampai dengan satu minggu.

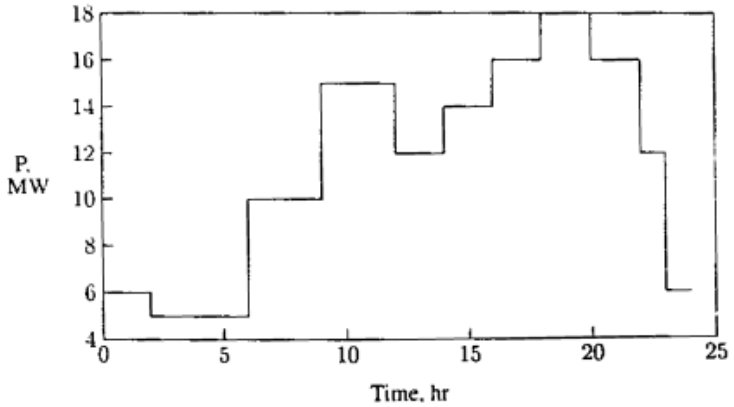
Berdasarkan daerah beban dapat dibagi menjadi:

1. Beban Industri, industri yang besar memungkinkan untuk mendapatkkan suplai daya dari jaringan subtransmisi dan industri kecil mendapatkan pelayanan daya dari jaringan distribusi primer. Beban industri merupakan beban gabungan yang memiliki fungsi tegangan dan frekuensi
2. Beban Komersial dan perumahan, beban jenis ini memiliki frekuensi yang relaltif tetap dan konsumsi daya reaktif yang kecil diabaikan.

Masing-masing sektor tersebut memiliki karakteristik beban yang berbeda, dikarenakan tiap konsumen disetiap sektor memiliki pola konsumsi yang berbeda.

Kurva beban secara sederhana dapat diartikan sebagai kurva yang menggambarkan penggunaan beban (listrik) dalam satuan waktu, entah

itu dalam selang waktu hari, minggu, atau tahun. Akan tetapi penggunaan kurva yang paling umum adalah kurva beban harian.




**Gambar 2.2** Kurva beban harian [4]

## 2.2 Karakteristik Pembangkit Thermal

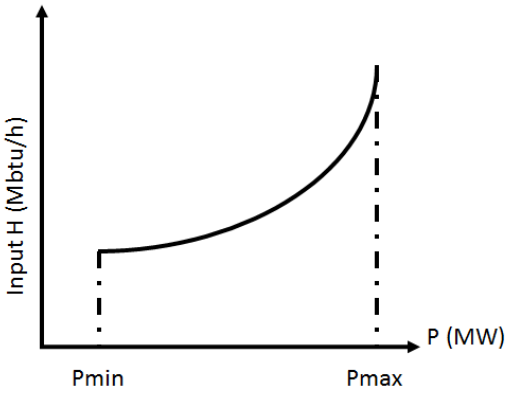
Ada banyak parameter dalam analisis pengaturan operasi sistem tenaga. Hal yang paling mendasar dalam masalah operasi ekonomis adalah karakteristik *input-output* pada pembangkit *thermal*. Untuk menggambarkan karakteristik *input-output*, *input* merepresentasikan sebagai masukan total yang diukur dalam satuan biaya/jam dan *output* merupakan daya keluaran listrik yang disediakan oleh sistem pembangkit tenaga listrik.Dalam menggambarkan karakteristik unit turbin uap, akan menggunakan *termionologi* (2.1) dan (2.2) sebagai berikut:

$$H = \frac{Mbtu}{jam} \tag{2.1}$$

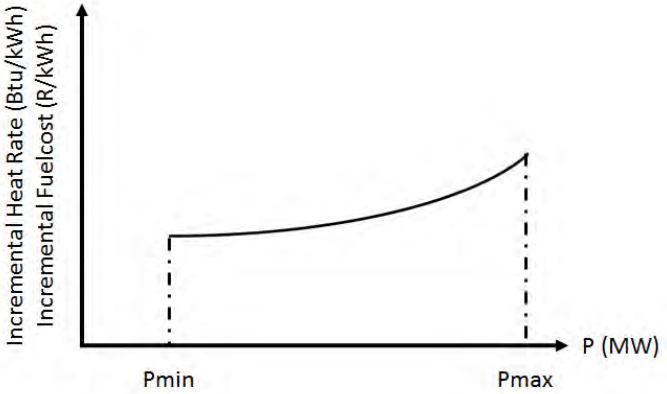
$$F = \frac{R}{jam} \tag{2.2}$$

H dapat dinyatakan sebagai energi panas yang dibutuhkan tiap jam dan F dinyatakan sebagai biaya tiap jam. Ada kalanya R/jam biaya operasional suatu unit terdiri dari biaya operasional dan biaya pemeliharaan. Biaya karyawan akan dimasukkan sebagai bagian dari biaya operasi jika biaya ini dapat digambarkan secara langsung sebagai fungsi dari *output* unit. *Output* dari unit pembangkit dinyatakan dengan P

dalam Megawatt. Untuk lebih jelasnya dapat dilihat pada Gambar 2.3 dan Gambar 2.4




**Gambar 2.3** Kurva input-output pembangkit thermal




**Gambar 2.4** Kurva incremental pembangkit thermal

Karakteristik *input-output* dari unit pembangkit *thermal* yang ideal, digambarkan sebagai kurva nonlinear yang kontinyu. Data karakteristik input output diperoleh dari perhitungan desain atau. Pembangkit *thermal* mempunyai batas operasi *minimum* (Pmin) dan *maxsimum* (Pmax). Batasan beban minimum biasanya disebabkan oleh kestabilan

pembakaran dan masalah desain generator. Pada umumnya unit pembangkit thermal tidak dapat beroperasi dibawah 30% dari kapasitas desain.

## 2.3 Economic Dispatch (ED)

Tingkat efisiensi dalam operasi optimalisasi ekonomi dan perencanaan daya pembangkitan listrik akan selalu menjadi bagian penting dalam perindustrian listrik. Oleh karena itu diperlukan perhitungan khusus akan pengiriman daya kepada para konsumen tenaga listrik sehinnga perusahaan pemasok listrik tidak mengalami kerugian.

Tujuan utama dari *Economic Dispatch* (ED) adalah untuk menentukan kombinasi daya *output* yang minimal dari setiap unit pembangkit, dengan meminimalkan total biaya bahan bakar, sementara dapat memenuhi kebutuhan baban para konsumen. Pengoptimalan permasalahan ED sangat penting untuk melakukan perkiraan jangka panjang dalam sistem tenaga listrik, penentuan porsi biaya, dan pemodelan managemen operasi tenaga listrik pada pembangkit.

Pembangkitan listrik memiliki tiga komponen biaya utama, antara lain biaya pembangunan, biaya kepemilikan, biaya operasionla. Biaya operasional merupakan biaya yang berkaitan langsung dengan keuntungan penjalanan produksi. Hal ini dikarenakan biaya operasional berhubungan langsung dengan managemen pembangkitan daya liatrik.

Salah satu bagian yang paling penting dalam biaya operasional adalah biaya bahan bakar (*fuelcost*). Pada setiap unit pembangkitan nilai yang berbeda tergantung dari jenis bahan bakar yang digunakan dalam pembangkitan. Nilai dari *fuelcost* sangat mempengaruhi fungsi biaya yang didapat. Secara umum nilai dari *fuelcost* dapat dinyatakan dalam persamaan (2.3) berikut.

$$fuelcost = \frac{R}{Mbtu} \tag{2.3}$$

*Fuelcost* merupakan harga persatuan panas dari bahan bakar, atau dapat dinyatakan sebagai konversi satuan panas ke satuan mata uang.

Pengaruh nilai *fuelcost* terhadap fungsi biaya dalam dilihat dalam persaman objektif ED berikut,

$$Hi(Pi) = aiPi^2 + biPi + ci \tag{2.4}$$

$$Fi(Pi) = Hi(Pi) \times fuelcost\ i \tag{2.5}$$

$$Ftotal = min \sum_{i=1}^{n} Fi(Pi) \quad (2.6)$$

Dimana
$n$ : jumlah generator

Dengan terhubungnya banyak unit pembangkit dalam sebuah sistem interkoneksi memberikan kemungkinan pengaturan pembangkitan yang lebih kecil untuk setiap unit.

*Equality Constrain* merupakan batasan yang merepresentasikan keseimbangan daya dalam sistem. Fungsi persamaan pada ED dinyatakan dalam persamaan,

$$\sum_{i=1}^{n} Pi = Pload + Ploss, n = jumlah\ generator \quad (2.7)$$

*Inequality Constrain* merupakan batasan yang merepresentasikan kapasitas daya dari pembangkit. Pada ED fungsi pertidaksamaan dinyatakan dalam persamaan (2.8) berikut.

$$Pi\ min \leq Pi \leq Pi\ max \quad (2.8)$$

Jika batasan *minimum* memiliki nilai seperti yang didapatkan pada persamaan (2.9) maka akan didapatkan solusi (2.10).

$$Pi \leq Pi\ min \quad (2.9)$$

$$Pi = Pi\ min \quad (2.10)$$

Jika batasan *maximum* memiliki nilai seperti yang didapatkan pada persamaan (2.11) maka akan didapatkan solusi (2.12).

$$Pi \geq Pi\ max \quad (2.11)$$

$$Pi = Pi\ max \quad (2.12)$$

## 2.4 Dynamic Economic Dispatch (DED)

*Dynamic Economic Dispatch* merupakan *Economic Dispatch* yang diperhitungkan dalam keadaan beban secara real time yang terus menerus berubah. Bentuk dari DED dapat dilihat pada persamaan objektif :

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci \quad (2.13)$$

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i \quad (2.14)$$

$$Ftotal = min \sum_{t=1}^{T} \sum_{i=1}^{n} Fi(Pi(t)) \quad (2.15)$$

Dimana
$n$ : jumlah generator
$T$ : total waktu dalam jam

*Equality constrain*

$$\sum_{i=1}^{T} Pi(t) = Pload,(t) + Ploss(t) \quad (2.16)$$

*Inequality constrain*

$$Pi\ min \leq Pi(t) \leq Pi\ max \quad (2.17)$$

## 2.5 Ramp-rate

Permintaan konsumsi listrik terus berubah, membuat variasi dan ketidakpastian karakteristik yang melekat pada sistem listrik. Akan tetapi generator sebagai penghasil listrik memiliki batasan tersendiri untuk menghasilkan daya yang optimal sesuai dengan perubahan beban. Dalam hal ini *ramp-rate* perlu diperhatikan sebagi batasan baru pembangkitan setiap unit generator.

*Ramp-rate* merupakan kemampuan generator untuk melakukan peningkatan (*up-rate*) atau penurunan (*down-rate*) generasi. Setiap unit pembangkit memiliki karakteristik yang berbeda, sehingga membutuhkan fungsi tertentu untuk mendapatkan hasil yang optimal.

Fungsi *ramp-rate* dapat dilihat dalam persamaan:

$$Pi(t-1) - Pi(t) \leq DRi \quad (2.18)$$

$$Pi(t) - Pi(t-1) \leq URi \quad (2.19)$$

Sehingga jika persamaan (2.18) dan (2.19) digabungkan akan mendapatkan batasan baru, yaitu:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1) \quad (2.20)$$

Nilai dari batasan *ramp-rate* tersebut tetap harus memenuhi batasan DED pada Tugas Akhir ini. Sehingga didapatkan syarat baru unuk perhitungan optimasi pada beban periode berikutnya.

Jika batasan *minimum ramp-rate* memiliki nilai yang didapatkan pada peramaan (2.21) maka akan didapatkan solusi (2.22).

$$Pi(t-1) - DRi < Pi\,min \tag{2.21}$$

$$Pi(t-1) - DRi = Pi\,min \tag{2.22}$$

Jika batasan *minimum ramp-rate* memiliki nilai yang didapatkan pada peramaan (2.23) maka akan didapatkan solusi (2.24)

$$URi + Pi(t-1) > Pi\,max \tag{2.23}$$

$$URi + Pi(t-1) = Pi\,max \tag{2.24}$$

## 2.6 Loss Formula

Dalam sistem tenaga, kerugian transmisi merupakan kehilangan daya yang harus ditanggung oleh unit pembangkitan. Sehingga daya yang hilang pada kerugian transmisi akan menjadi beban tambahan pada sistem tenaga.

Rugi-rugi dalam jaringan transmisi sistem tenaga akan mejadi sebuah fungsi pembangkitan. Berdasarkan formula rugi-rugi yang konstan, fungsi didapatkan dalam persamaan quadratik yang diselesaikan dengan mengetahui *B coefficient* [6], yang dapat dikenal sebagai *Bloss matrix*.

*Bloss matrix* formula sudah diperkenalkan sejal 1950 sebagai metode untuk mendapatkan nilai rugi-rugi dan *incremental loss* dalam perhitungan [7]. Dimana persamaan daya loss secara umum dapat dilihat pada persamaan:

$$\begin{aligned} Ploss = [P1 \quad \dots \quad Pn] \begin{bmatrix} B11 & \cdots & B1j \\ \vdots & \ddots & \vdots \\ Bi1 & \cdots & Bij \end{bmatrix} \begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} \\ + [Bo1 \quad \dots \quad Boi] \begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + Boo \end{aligned} \tag{2.25}$$

Sehingga pada persamaan yang didapatkan pad (2.25) dapat disederhanakan menjadi persamaan (2.26) berikut:

$$Ploss = P^T[B]P + Bo^T P + Boo \tag{2.26}$$

Dimana,

$P$ : *vector* semua generator (MW)

$[B]$ : *matrix* persegi dari dimensi yang sama dengan P

$Bo$ : vector dengan panjang yang sama dengan P

$Boo$ : koefisien konstan

Nilai pada Bij secara umum merepresentasikan koefisien rugi-rugi dengan persamaan umum:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo \qquad (2.27)$$

Nilai dari rugi-rugi daya biasanya berkisar antara 20 % hingga 30 % dari jumlah semua total beban.

Ketika nilai rugo-rugi diperhitungkan maka *penalty factor* di setiap unit akan berbeda. Tidak seperti ketika mengitung nilai optimum pembangkitan tanpa menggunakan rugi-rugi, dimana nilai *penalty factor* dianggap 1 (satu). Persamaan *penalty factor* dapat dilihat pada persaman (2.28) berikut:

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}} \qquad (2.28)$$

Dimana memiliki hubungan dengan *incremental loss* yang dituliskan dalam persamaan (2.29) berikut:

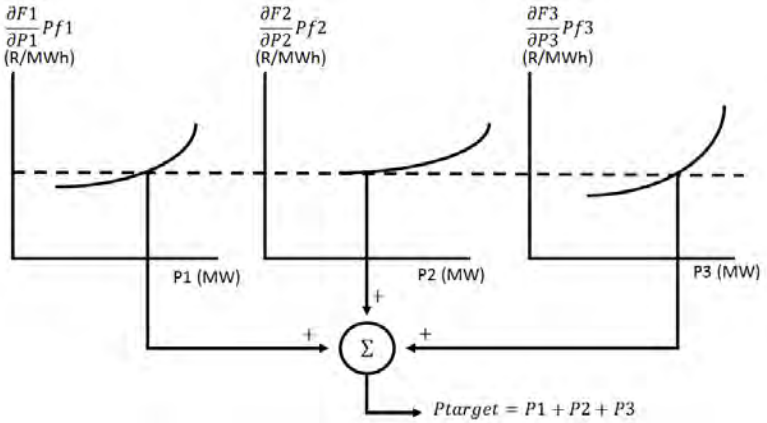
$$incremental\ loss = \frac{\partial Ploss}{\partial Pi} \qquad (2.29)$$

Nilai dari *penalty factor* akan mempengaruhi nilai lambda dalam perhitungan Tugas Akhir ini. Pengaruh dari *penalty factor* akan terlihat dalam persamaan yang akan dijelaskan dalam Subbab 2.7**.**

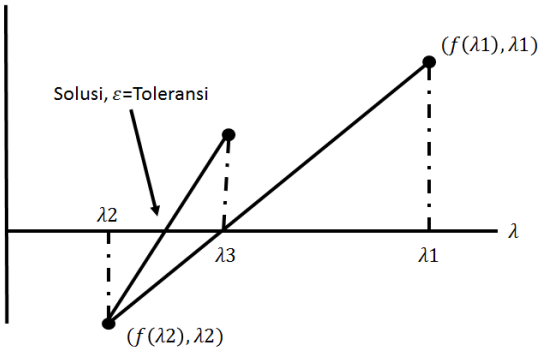
## 2.7 Iterasi Lambda

Iterasi Lambda digunakan pada Tugas Akhir ini untuk menyeleasaikan persamaan *Dynamic Economic Dispatch*.

Pada metode iterasi lambda, nilai lambda pertama akan ditentukan terlebih dahulu. Tentunya nilai dari lambda petama bukanlah hasil yang benar. Ketika niali total dari $P1 + P2 + P3 + \cdots Pi < Ptarget$ maka nilai $\lambda$ untuk itersi berikutnya akan bertambah lebih besar dari nilai $\lambda$ sebelumnya. Dan sebaliknya, jika nilaii total $P1 + P2 + P3 + \cdots Pi > Ptarget$ maka nilai lambda untuk iterasi berikutnya akan lebih kecil daripada nilai dari lambda sebelumnya. Proses ini akan melakukan itersi nilai lambda hingga mendapatkan hasil dimana $P1 + P2 + P3 + \cdots Pi = Ptarget$. Seperti yang dijelaskan dalam Gambar 2.5 dan Gambar 2.6




**Gambar 2.5** Grafik penyelesaian iterasi lambda




**Gambar 2.6** Proyeksi lambda

Nilai lambda dapat dilihat pada persamaan lagrangian:

$$\frac{\partial Fi}{\partial Pi} = \lambda\left(1 - \frac{\partial Ploss}{\partial Pi}\right) \tag{2.30}$$

$$\lambda = \frac{\partial Fi}{\partial Pi}\left(\frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}\right) \tag{2.31}$$

$$\lambda = \frac{\partial Fi}{\partial Pi} Pfi \tag{2.32}$$

Menentuka nilai lambda awal dilihat dalam persamaan 2.23:

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right) \tag{2.33}$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right) \tag{2.34}$$

$$\lambda start = \frac{\lambda \max + \lambda min}{2} \tag{2.35}$$

Untuk melakukan iterasi maka diperlukan $\Delta\lambda$, yang dapat dilihat dalam persamaan:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} \tag{2.36}$$

$$Ptarget = Ploss + Pload \tag{2.37}$$

Jika nilai yang didaptkan dalam proses iterasi sama dalam peersamaan (2.38), maka solusi peerubahan nilai lambda untuk iterasi berikutnya adalah (2.39).

$$\sum_{i}^{n} Pi - Ptarget > 0, \qquad n = Jumlah\ Generator \tag{2.38}$$

$$\lambda = \lambda sebelum - \Delta\lambda \tag{2.39}$$

Jika nilai yang didaptkan dalam proses iterasi sama dalam peersamaan (2.40), maka solusi peerubahan nilai lambda untuk iterasi berikutnya adalah (2.41).

$$\sum_{i}^{n} Pi - Ptarget < 0, \qquad n = Jumlah\ Generator \tag{2.40}$$

$$\lambda = \lambda sebelum + \Delta\lambda \tag{2.41}$$

Nilai lambda akan terus berubah hingga mendapatkan nilai dalam persamaan (2.42) berikut,

$$\sum_{i}^{n} Pi - Ptarget = 0, \qquad n = Jumlah\ Generator \tag{2.42}$$

Nilai $\lambda$ pada iterasi kedua biasanya bernilai 10 % lebih besar dari nilai $\lambda$ pertama, atau 10 % kurang dari nilai $\lambda$ pertama tergantug dari hasil perhitungan error pada iterasi tertentu.

Pada Tugas Akhir ini nilai *error* ($\varepsilon$) ditentukan sebesar 0.01. Ketika diimplementasikan ke dalam persamaan, maka nilai lambda akan berhenti melakukan iterasi hingga mendapatkan nilai,

$$\sum_{i}^{n} Pi - Ptarget = \varepsilon, \qquad n = Jumlah\ Generator \quad (2.43)$$

Dimana setiap nilai *Pi* harus memenuhi syarat *inequality constrain* dan batasan *ramp-rate* yang digunakan untuk penyelesaian masalah DED pada Tugas Akhir ini.

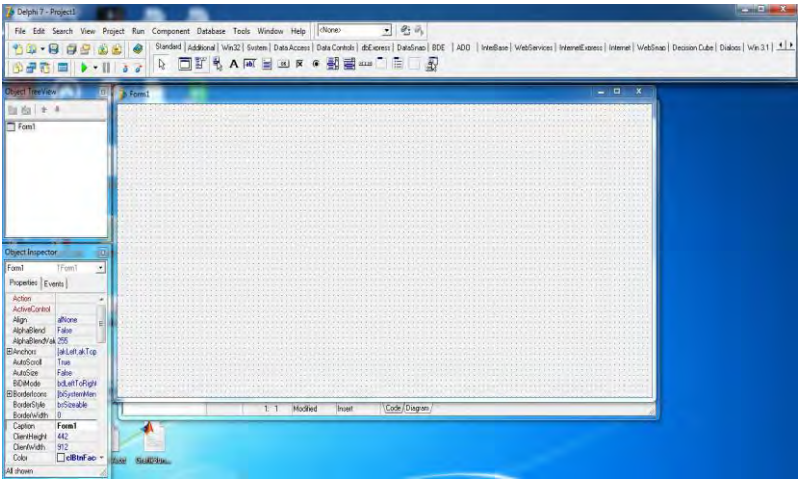
## 2.8 Delphi

Delphi merupakan bahasa pemrograman berbasis *Windows* yang menyediakan fasilitas pembuatan aplikasi *visual*. Pada Tugas Akhir ini menggunakan Delphi untuk mengaplikasikan formula DED yang ada. Dengan menghasilkan aplikasi perhitungan DED yang memiliki *interface* yang mudah dipahami, dengan batasan masalah yang telah ditentukan. Delphi sendirinya memiliki beberapa kelebihan diantaranya:

1. Kemudahan penyusunan *User Interface*, Delphi berkomitmen untuk menjadi *Rapid Aplication Development* (RAD). Maksudnya adalah bagaimana mempercepat perkembangan aplikasi.
2. Bahasa *Object Pascal*, merupakan salah satu varian dari bahas pascal dengan sejumlah penambahan, terutama terkait dengan dengan konsep *Object Oriented Programing* (OOP). Dengan salah satu kelebihan bahasa Pascal yang mudah dipahami dan tidak terlalu kompleks.
3. *Native Code*, hasil kompile Delphi adalah kode *native* untuk window 32. Ini berarti *file exe* yang dihasilkan oleh kompiler akan langsung dijalankan oleh mesin tanpa melalui *software* lain seperti *Virtual Machine* (VM). Secara umum *native code* lebih cepat daripada penjalan program dengan VM, hal ini dikarenakan Delphi menggunakan *installer* sederhana. Hasil dari Delphi adalah *file exe* tunggal, tanpa perlu file-file lainnya.

Ketika memulai Delphi, maka akan ditempatkan ke dalam *Integrated Development Environment* (IDE). IDE ini menyediakan semua

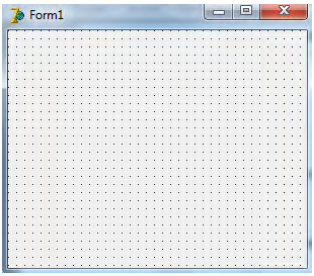
alat yang dibutuhkan dalam merancang, mengembangka, menguji, *debug*, dan penyebaran aplikasi dalam waktu yang singkat. Dalam hal ini IDE mencangkup semua alat yang diperlukan untuk memulai perancangan aplikasi seperti seperti yang terlihat pada Gambar 2.7 berikut.



**Gambar 2.7** Tampilan pengerjaan Delphi

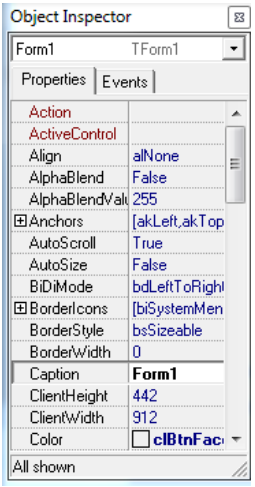
Beberapa bagian penting yang perlu diketahui dalam pemrograman dengan menggunakan Delphi diantaranya adalah:

1. *Form Designer* atau *Form*, merupakan jendela kosong yang digunakan untuk merancang suatu *User Interface* (UI) dalam perancangan aplikasi yang sedang dibuat. Dapat dilihat pada Gambar 2.8 berikut,



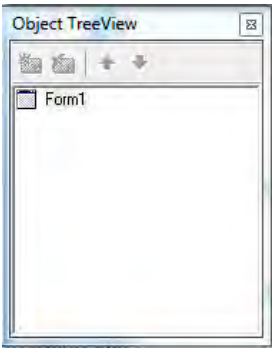

**Gambar 2.8** Tampilan *Form Designer*

2. *Object Inspector*, digunakan untuk mengatur dan memeriksa sekumpulan *property* yang berada di dalam *Form* untuk mendapatkan tampilan sesuai denganyang diinginkan. *Object Inspector* dapat dilihat pada Gambar 2.9.



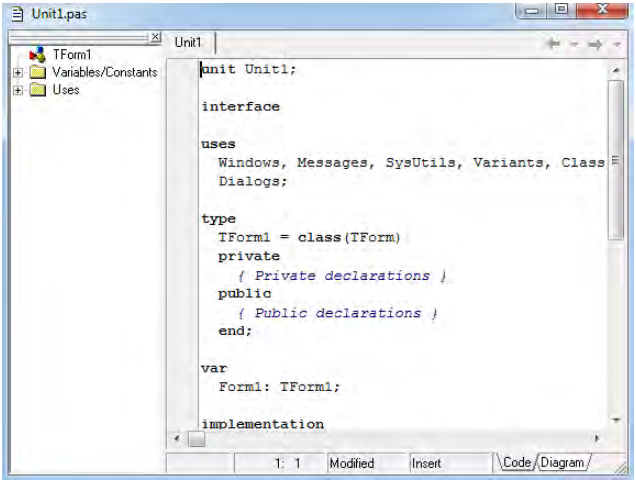

**Gambar 2.9** Tampilan *Object Inspector*

3. *Object TreeView*, digunkan untuk menampilkan dan mengubah hubungan antar komponen dalam *Form*. Tampilan dapat dilihat pada Gambar 2.10.




**Gambar 2.10** Tampilan *Object TreeView*

4. *Code Editor*, merupakan tempat penulisan dan edititing logika pemrograman yang sedang dibuat. Dapat dilihat pada Gambar 2.11.



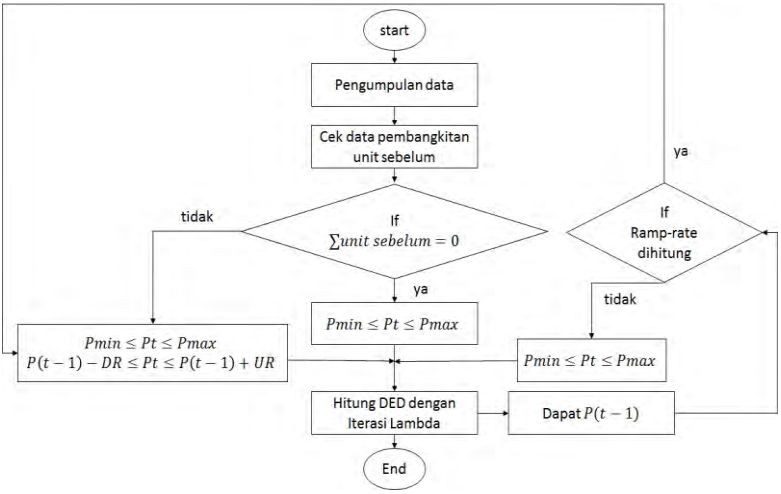

**Gambar 2.11** Tampilan *Code Editor*

*Halaman ini Sengaja Dikosongkan*

# BAB 3
# IMPLEMENTASI DYNAMIC ECONOMIC DISPATCH MENGGUNKAN ITERASI LAMBDA DENGAN MEMPERHATIKAN RAMP-RATE

Dalam Tugas Akhir ini, iterasi lambda digunakan untuk menyelesaikan permasalahan dalam perhitungan *Economic Dispatch* (ED) yang kemudian akan diterapkan pada *Dynamic Economic Dispatch* (DED). Hal ini dikarenakan perubahan batasan yang akan terjadi dalam persaaman perhitungan iterasi lambda pada DED. Pengaruh dari syarat batasan *ramp-rate* yang akan mempengaruhi perubahan batasan pembangkitan daya *maximum* dan daya *minimum* untuk mendapatkan hasil pembangkitan yang optimal. Sehingga dengan didapatkannya nilai kombinasi pembangkitan yang *minimum* akan didapatkan nilai biaya *minimum*. Pengolahan data dan simulasi menggunakan aplikasi perhitungan yang dibuat dengan menggunakan Delphi untuk membantu penyelesaian Tugas Akhir ini. Alur penyelesaian serta penerapan metode akan diterapkan pada subbab 3.1 berikut.

## 3.1 Alogaritma DED




**Gambar 3.1** Flowcart penerapan DED pada Delphi

Alur dari penggunaan aplikasi perhitungan DED dimulai dengan mengumpulkan semua data yang dibutuhkan. Mulai dari jumlah unit, koefisien tiap orde untuk *Incremental Heat Rate*, *fuelcost*, data batasan *maximum* dan *minimum* tiap unit, batasan *ramp-rate*, dan nilai pembangkitan unit pada jam sebelum. Setelah itu menentukan berapa beban yang ingin diperhitungkan dan mengisi data beban pada setiap jam. Perhitungan akan mengggunakan metode iterasi lambda untuk menentukan pembangkitan setiap unit. Hasil pembangkitan akan dimasukan ke dalam persamaan fungsi biaya untuk mendapatkan nilai biaya pada beban setiap jam yang telah ditentukan.

## 3.2 Inisiasi Persamaan Objektif dan Constrain DED

Seperti yang telah dijelaskan di dalam bab 2 sebelumnya, DED memiliki persamaan objektif yang dipergunkan untuk mendapatkan nilai pembangkitan dan biaya ketika dikalikan dengan *fuelcost*. Persamaan objektif DED dapat dilihat dalam persaman berikut:

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci \tag{3.1}$$

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i \tag{3.2}$$

Dan dengan persamaan *inequlity constrain* yang telah di pengaruhi oleh *ramp-rate* dapat dilihat dalam persamaan:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1) \tag{3.3}$$

Dalam program perhitungan Delphi juga diberikan persamaan rugi-rugi yang telah ditentukan nilainya berupa *Bloss matrix* yang akan diiterasikan hingga mencapai nilai rugi-rugi yang tidak berubah. Persamaan rugi-rugi dalam DED dapat dilihat dalam persamaan:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo \tag{3.4}$$

Sehingga dengan adanya penambahan nilai rugi-rugi akan menjadi beban tambahan dalam sistem operasi. Hal ini akan menghasilkan persamaan *equality constrain*, yaitu:

$$Ptarget = Ploss + Pload \tag{3.5}$$

$$\sum_{i}^{n} Pi - Ptarget = \varepsilon, \qquad n = Jumlah\ Generator \tag{3.6}$$

## 3.3 Agumen Input DED

Dari semua persaman dibuat argument inputan dalam bahasa Delphi sebagai masukan awal dalam perhitungan DED menggunakan iterasi lambda nantinya. Argumen yang digunakan dapat dilihat pada Tabel 3.1 sebagai berikut:

**Tabel 3.1** Argument input

| Argumen | Keterangan |
|---|---|
| Coeff[i] | Sebagai inputan awal dari nilai koefisien A, B, C dalam persamaan $$Hi(Pi(t)) = aiPi(t)^2 + biPi(t) + ci$$ |
| Unitmax[i] | Sebagai inputan awal dari batas *maximum* pembangkitan unit (Pmax) |
| Unitmin[i] | Sebagai inputan awal dari batasan *minimum* pembangititan unit (Pmin) |
| UR[i] | Sebagai inputan awal dari batasan atas *ramp-rate* (UR) |
| DR[i] | Sebagai inputan awal dari batasan bawah *ramp-rate* (DR) |
| Fuelcost[i] | Sebagai inputan awal nilai dari *fuelcost* yang digunakan untuk persaman $$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$$ |
| Unitsebelum [i] | Sebagai inputan dan identifikasi hasil *Dynamic Economic Dispatch* pada jam sebelumnya. Hasil nilai yang didapatkan dari perhitungan akan digunakan untuk menentukan *inequality constrain* yang baru untuk perhitungan DED dengan beban pada periode berikutnya $$Pi(t-1) - DRi \leq Pi(t)$$ $$Pi(t) \leq URi + Pi(t-1)$$ |
| Loadjam[i] | Sebagai inputan nilai beban ($Pload$) di setiap periode |
| B00[i], B0[i], B[i,i], B[i,j] | Sebagai inputan dari nilai koefisien pada *Bloss matrix* dalam persamaan $$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo$$ |

## 3.3 Sintaksis DED

Sintaksis program merupakan perintah yang digunakan untuk melakukan pemanggilan program dengan argument input yang telah kita tentukan. Sintaksis yang digunakan dalam Delphi dilihat dalam Tabel 3.2 sebagai berikut:

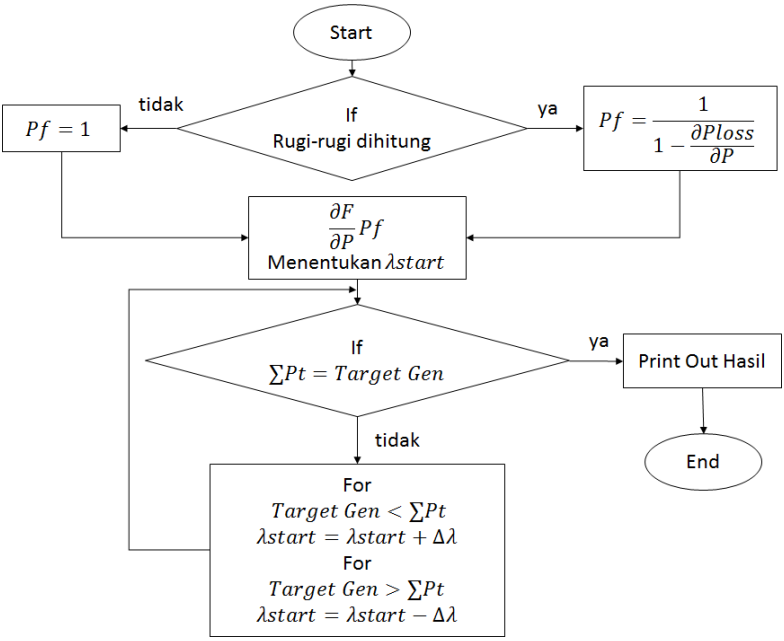
**Tabel 3.2** Sintaksis

| Sintaksis | Keterangan |
|---|---|
| datadump | Digunakan sebagai memasukkan semua data awal perhitungan di setiap periode |
| ihr_ftn | Sebagai inisiasi turunan pertama persamaan $Hi(Pi(t))$ yang nantinya akan dikalikan dengan *fuelcost*, sehingga didapatkan inisiasi $\frac{\partial Fi}{\partial Pi}$ |
| invers_ihr_ftn | Mencari nilai pembangkitan setiap unit ketika didapatkan nilai lambda $\frac{\partial Fi}{\partial Pi} = \lambda\left(1 - \frac{\partial Ploss}{\partial Pi}\right)$ |
| lambda_search_ dispatch | Sebagai prosedure penjalanan metode iterasi lambda, yaitu dengan menentukan nilai $\lambda start$ dan $\Delta\lambda$ untuk proses iterasi. Hasil akhir bertujuan untuk mendapatkan jumlah nilai pembangkitan $\sum_{i}^{n} Pi - Ptarget = \varepsilon$ |
| loss_matrix_ftn | Mencari besaran nilai rugi-rugi dari persaman *Bloss matrix* |
| prod_cost | Mendapatkan nilai biaya pembangkitan setiap unit setelah mendapatkan nilai pembangkitan yang optimal dari proses iterasi lambda |

## 3.4 DED Meggunakan Metode Iterasi Lambda

Sebelum melakukan pemrograman yang kan diimplementasikan kedalam bahasa pemrograman Delphi yaitu bahasa *Pascal*, maka dibuat terlebih dahulu alur perhitungan DED.

Pada Tugas Akhir ini alur perhitungan DED menggunakan metode iterasi lambda, dari alur perhitungan didapatkan alur *flowcart* untuk dimplementasikan ke dalam Dlephi yang dapat digambarkan secara umum dalam Gambar 3.2 berikut,




**Gambar 3.2** Flowcart iterasi lambda

Pada proses iterasi dibutuhkan nilai lambda awal dengan menggunakan prsamaan (3.7) dan perubahan lambda untuk proses iterasi pada persamaan (3.8).

$$\lambda start = \frac{\lambda max - \lambda min}{2} \tag{3.7}$$

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} \tag{3.8}$$

Dimana nilai dari $\lambda max$ dapat ditentukan dengan menggunkan persamaan (2.34) dan nilai $\lambda min$ dapat ditentukan dengan menggunakan persamaan (2.33).

Beasarnya nilai lambda yang digunakan sebagai proses iterasi pada metode ini adalah,

$$\lambda = \frac{\Delta\lambda}{2} \tag{3.9}$$

Hingga didapatkan nilai lambda yang sesuai. Dalam hal ini nilai lambda akan berhenti ketika *equality constrain* telah terpenuhi. dengan syarat daya pembangkitan setiap unit harus memenuhi batasan,

$$Pi\ min \leq Pi(t) \leq Pi\ max \tag{3.10}$$

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1) \tag{3.11}$$

Dimana,

$$\frac{\partial Fi}{\partial Pi} = \lambda\left(1 - \frac{\partial Ploss}{\partial Pi}\right) \tag{3.12}$$

Diberikan contoh perhitungan untuk memahami alur dari penggunaan metode iterasi lambda pada DED. Dapat dilihat pada contoh kasus ini, melakukan perhitungan DED menggunakan sistem dengan 3 unit generator. Dengan permintaan daya setiap periode berintervalkan waktu 1 jam. Contoh ini akan menjelaskan hasil perhitungan DED pada load 2 jam pertama, dengan data sebagai berikut:

**Tabel 3.3** Contoh karakteristik pembangkit

| | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| A | 0.00142 | 0.00194 | 0.0048 |
| B | 7.2 | 7.85 | 7.97 |
| C | 510 | 310 | 78 |
| Pmin | 150 | 100 | 50 |
| Pmax | 600 | 400 | 200 |
| UR | 10 | 30 | 20 |
| DR | 10 | 30 | 20 |
| Fuelcost | 1.1 | 1 | 1 |

**Tabel 3.4** Contoh beban

| Jam | 1 | 2 |
|---|---|---|
| Beban | 850 | 800 |

Berikut adalah hasil perhitungan DED yang didapatkan ketika tidak melihat hasil rugi-rugi daya.

Pertama DED akan melakukan perhitunga untuk beban pada jam ke 1 (satu) kemudian melakukan perhitungan pada jam ke 2 (dua). Seperti data yag digunakan pada Tabel 3.2,

JAM 1, Beban 850 MW

Nilai *heat rate* :

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci$$

$$H1 = 510 + 7.2P1 + 0.00142P1^2$$

$$H2 = 310 + 7.85P2 + 0.00194P2^2$$

$$H3 = 78 + 7.97P3 + 0.00482P3^2$$

Fungsi biaya :

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$$

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$

Batasan tiap unit untuk perhitungan jam ke 1 mrnggunkan batasam maximum dan minimum yang dimilki oleh pembangkit, sebab asumsi untuk melakukan perhitungann DED pada jam ke 1 adalah dimana keadaan setiap unit pembangkit sedang dipersiapkan untuk menyala,maka batasan utuk perhitungan DED jam ke 1:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P_1 \leq 600$$

$$100 \leq P_2 \leq 400$$

$$50 \leq P_3 \leq 200$$

Menetukan lambda *minimum* dan lambda *maximum*, dikarenakan nilia rugi-rugi tidak diperhitungkan maka nilai $Pf = 1$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.2380$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.8980$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.0680$$

Menentukan perubahan lambda (dicari untuk melakukan iterasi):

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.83$$

Melakukan metode iterasi lambda,

Yaitu dengan memasukkan nilai lambda yang didapat kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load jam ke 1 yang diinginkan sebesar 850 MW.

Hasil proses iterasi lambda,

$$\lambda_{iter1} = 9.0680 \ldots\ldots\ldots totalgen = 795.3$$

$$\lambda_{iter2} = 9.8980 \ldots\ldots\ldots totalgen = 1200$$

$$\lambda_{iter3} = 9.4830 \ldots\ldots\ldots totalgen = 1057.3$$

$$\lambda_{iter4} = 9.2755 \ldots\ldots\ldots totalgen = 936.7$$

.
.

$$\lambda_{iter17} = 9.1483 \dots \dots \dots totalgen = 850.0$$

$$P1 = \frac{\lambda - 7.92}{2 \times 0.001562} = \frac{9.1483 - 7.92}{2 \times 0.001562} = 393.2$$
$$150 \leq P1 \leq 600$$
$$P1 = 393.2$$

$$P2 = \frac{\lambda - 7.85}{2 \times 0.00194} = \frac{9.1483 - 7.85}{2 \times 0.00194} = 334.6$$
$$100 \leq P2 \leq 400$$
$$P2 = 334.6$$

$$P3 = \frac{\lambda - 7.97}{2 \times 0.00482} = \frac{9.1483 - 7.97}{2 \times 0.00482} = 112.2$$
$$50 \leq P3 \leq 200$$
$$P3 = 112.2$$

Maka didapatkan hasil DED pembangkitan tiap unit untuk beban jam ke 1 sebesar 850 MW adalah,

$$P1 = 393.2$$

$$P2 = 334.6$$

$$P3 = 122.2$$

Nilai biaya operasi didapatkan setelah semua nilai pembangkitan setiap unit telah diketahui yang kemudian dimasukkan ke dalam fungsi biaya, maka didapatkan biaya operasi setiap unit sebagai berikut

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F1 = 561 + 7.92(383.2) + 0.001562(383.2^2)$$
$$F1 = 3825.06 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$
$$F2 = 301 + 7.85(306.1) + 0.00194(306.1^2)$$
$$F2 = 2894.52 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(110.7) + 0.00482(110.7^2)$$
$$F3 = 1019.78 \text{ (\$/jam)}$$

JAM 2, beban 800 MW
Batasan tiap unit jam ke 2:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P1 \leq 600$$

$$100 \leq P2 \leq 400$$

$$50 \leq P3 \leq 200$$

Syarat *ramp-rate*:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1)$$

Sehingga didapatkan batasan unit baru dengan memperhatikan kedua batasan.
Batasan baru tiap unit jam ke 2:

$$383.17 \leq P1 \leq 403.17$$

$$304.61 \leq P2 \leq 364.61$$

$$102.23 \leq P3 \leq 142.23$$

Menetukan lambda *minimum* dan lambda maximum, dikarenakan nilia rugi-rugi tidak diperhitungkan maka nilai $Pf = 1$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.9555$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.3411$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.0680$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.1928$$

Melakukan metode iterasi lambda,

Dengan memasukkan nilai lambda yang baru diakibatakan oleh pengaruh *ramp-rate* yang didapat kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 800 MW.

Hasil proses iterasi lambda,

$$\lambda_{iter1} = 9.1483 \ldots\ldots\ldots totalgen = 850.0$$

$$\lambda_{iter2} = 8.9555 \ldots\ldots\ldots totalgen = 790.0$$

$$\lambda_{iter3} = 9.0519 \ldots\ldots\ldots totalgen = 805.2$$

$$\lambda_{iter4} = 9.0037 \ldots\ldots\ldots totalgen = 795.0$$

$$\vdots$$

$$\lambda_{iter14} = 9.0376 \ldots\ldots\ldots totalgen = 800.0$$

$$P1 = \frac{\lambda - 7.92}{2 \times 0.001562} = \frac{9.0376 - 7.92}{2 \times 0.001562} = 357.8$$

$$383.17 \leq P_1 \leq 403.17$$

$$P1 = 383.2$$

$$P2 = \frac{\lambda - 7.85}{2 \times 0.00194} = \frac{9.0376 - 7.85}{2 \times 0.00194} = 306.1$$
$$304.61 \leq P_2 \leq 364.61$$
$$P2 = 306.1$$

$$P3 = \frac{\lambda - 7.97}{2 \times 0.00482} = \frac{9.0376 - 7.97}{2 \times 0.00482} = 110.7$$
$$102.23 \leq P_3 \leq 142.23$$
$$P3 = 110.7$$

Maka didapatkan hasil DED pembangkitan tiap unit beban jam ke 2 sebesar 800 MW adalah,

$$P1 = 383.2$$

$$P2 = 306.1$$

$$P3 = 110.7$$

Nilai biaya operasi

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F1 = 561 + 7.92(383.2) + 0.001562(383.2^2)$$
$$F1 = 3825.06 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$
$$F2 = 301 + 7.85(306.1) + 0.00194(306.1^2)$$
$$F2 = 2894.52 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(110.7) + 0.00482(110.7^2)$$
$$F3 = 1019.78 \text{ (\$/jam)}$$

Ketika nilai rugi-rugi diperhitungkan, Hasil pembangkitan pada tiap periode akan berubah menjadi lebih besar dari sebelumnya. Hal ini dikarenakan rugi-rugi akan menjadi daya tambahan sebagai beban. Nilai lambda pada proses iterasi akan berubah dikarenakn pengaruh dari $Pf \neq 1$. Jika dibandingkan dengan proses iterasi yang tidak melihat rugi-rugi daya, dimana nilai $Pf = 1$.

Nilai Pf didapatkan dalam persamaan berikut,

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}} \tag{3.13}$$

Dengan adanya besaran nilai rugi-rugi daya akan mengakibatkan penambahan beban. Sehingga pembangkitan akan berubah menjadi lebih besar dikarenakan beban yang bertambah.

Diberikan contoh kasus dengan menggunkan data tiga unit pembnagkit yang telah diberikan pada Tabel 3.1 dan data beban pada Tabel 3.2. Persaman rugi- rugi dinyatakan dalam sebuah *Bloss matrix* yang telah ditentukan pada data berikut,

**Tabel 3.5** Contoh *Bloss matrix*

| Bij | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.00003 | 0 | 0 |
| 2 | 0 | 0.00009 | 0 |
| 3 | 0 | 0 | 0.00012 |

Berikut adalah perhitungan DED yang didapatkan dengan memperhatikan nilai rugi-rugi daya yang didapatkan,

JAM 1, beban 850 MW

Nilai *heat rate* :

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci$$

$$H1 = 510 + 7.2P1 + 0.00142P1^2$$

$$H2 = 310 + 7.85P2 + 0.00194P2^2$$

$$H3 = 78 + 7.97P3 + 0.00482P3^2$$

Fungsi biaya :

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$$

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$

Batasan tiap unit jam ke 1:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P1 \leq 600$$

$$100 \leq P2 \leq 400$$

$$50 \leq P3 \leq 200$$

Besarnya nilai rugi-rugi daya pada iterasi 1:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo$$

$$Ploss = 0.00003P1^2 + 0.00009P2^2 + 0.00012P3^2$$

$$P1 = \frac{I50 + 600}{2} = 375$$

$$P2 = \frac{100 + 400}{2} = 250$$

$$P3 = \frac{50 + 200}{2} = 125$$

$$Ploss = 11.7$$

Menentukan besar beban baru akibat penamnhan nilai rugi-rugi pada iterasi 1,

$$Ptarget = Ploss + Pload$$
$$Ptarget = 11.7 + 850 = 861.7$$

Ketika nilai rugi-rugi diperhitungkan, penetukan nilai lambda min dan lambda max dipengaruhi oleh nilai $Pf$. Hal ini dikarenakan nilai dari $Pf \neq 1$, sehingga nilai lambda min dan max akan didapatkan,

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(375) = 0.0225$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(250) = 0.045$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(125) = 0.03$$

$$Pfi = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1 - 0.0225}$$

$$Pf2 = \frac{1}{1 - 0.045}$$

$$Pf3 = \frac{1}{1 - 0.03}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.5817$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 10.2041$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.3929$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.8112$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 1,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 861.7 MW.

$$\lambda_{iter1} = 9.3929 \dots \dots \dots totalgen = 810.92$$

$$\lambda_{iter2} = 10.2041 \dots \dots \dots totalgen = 1200$$

$$\lambda_{iter3} = 9.7985 \dots \dots \dots totalgen = 1078.48$$

$$\lambda_{iter4} = 9.5957 \dots \dots \dots totalgen = 944.7$$

$$\vdots$$

$$\lambda_{iter15} = 9.4699 \dots \dots \dots totalgen = 861.7$$

$$P1 = \frac{\lambda(1 - 0.0225) - 7.92}{2 \times 0.001562} = \frac{9.4699(1 - 0.0225) - 7.92}{2 \times 0.001562}$$

$$150 \leq P1 \leq 600$$

$$P1 = 427.9$$

$$P2 = \frac{\lambda(1 - 0.045) - 7.85}{2 \times 0.00194} = \frac{9.4699(1 - 0.045) - 7.85}{2 \times 0.00194}$$

$$100 \leq P2 \leq 400$$

$$P2 = 307.67$$

$$P3 = \frac{\lambda(1-0.03)-7.97}{2\times 0.00482} = \frac{9.4699(1-0.03)-7.97}{2\times 0.00482}$$
$$50 \le P3 \le 200$$
$$P3 = 126.12$$

Nilai $P1, P2, P3$ yang didapatkan kemudian digunakan untuk mencari rugi-rugi iterasi 2

$$PLoss = 0.00003(427.9^2) + 0.00009(307.67^2) + 0.00012(126.12.8^2)$$

$$Ploss = 15.9$$

Menentukan besar beban baru akibat penamnhan nilia rugi-rugi pada iterasi 2,

$$Ptarget = Ploss + Pload$$

$$Ptarget = 15.9 + 850 = 865.9$$

Menentukan lambda *minimum* dan lambda *maximum*:

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(427.9) = 0.02685$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(307.67) = 0.068103$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(126.12) = 0.03356$$

$$Pfi = \frac{1}{1-\frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1-0.02685}$$

$$Pf2 = \frac{1}{1-0.068103}$$

$$Pf3 = \frac{1}{1 - 0.03356}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.6097$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 10.2070$$

Menentukan lambda start:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.4083$$

Menentukan delta lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.7986$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 2,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 865.9 MW.

$$\lambda_{iter1} = 9.4083 \ldots\ldots\ldots totalgen = 786.08$$

$$\lambda_{iter2} = 10.2070 \ldots\ldots\ldots totalgen = 1200$$

$$\lambda_{iter3} = 9.8076 \ldots\ldots\ldots totalgen = 1048.015$$
$$\lambda_{iter4} = 9.5081 \ldots\ldots\ldots totalgen = 851.566$$

.

.

$$\lambda_{iter16} = 9.5300 \dots \dots \dots totalgen = 865.9$$

$$P1 = \frac{\lambda(1 - 0.02567) - 7.92}{2 \times 0.001562} = \frac{9.53(1 - 0.02567) - 7.92}{2 \times 0.001562}$$
$$150 \leq P1 \leq 600$$
$$P1 = 437.1$$

$$P2 = \frac{\lambda(1 - 0.05539) - 7.85}{2 \times 0.00194} = \frac{9.53(1 - 0.05539) - 7.85}{2 \times 0.00194}$$
$$100 \leq P2 \leq 400$$
$$P2 = 278.9$$

$$P3 = \frac{9.53(1 - 0.03027) - 7.97}{2 \times 0.00482} = \frac{9.53(1 - 0.03027) - 7.97}{2 \times 0.00482}$$
$$50 \leq P3 \leq 200$$
$$P3 = 137.1$$

Nilai $P1, P2, P3$ yang didapatkan pada iterasi ke 2, kemudian digunkan untuk mencari rugi-rugi iterasi ke 3. Hal ini kan terus berlangsung hingga didapatkan nilai $P1, P2, P3$ pada rugi-rugi iterasi ke 11. Nilai rugi-rugi yang didapatkan pada iterasi ke 11 adalah sebesar 15.8 Sehingga total load yang diinginkan akan sebesar 865.8.

Didapatkan nilai lambda yang telah diiterasi untuk rugi-rugi pada iterasi ke 11 sebesar 9.5284, sehingga nilai yang pembangkitan setiap unit yang didapatkan sebesar

$$P1 = 435.2$$

$$P2 = 300.0$$

$$P3 = 130.7$$

Sehingga nilai biaya pada periode pertama didapatkan

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F_1 = 561 + 7.92(435.2) + 0.001562(435.2^2)$$
$$F1 = 4303.59 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$

$$F2 = 301 + 7.85(300) + 0.00194(300^2)$$
$$F2 = 2839.31 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(130.7) + 0.00482(130.7^2)$$
$$F3 = 1201.65 \text{ (\$/jam)}$$

JAM 2, beban 800 MW
Batasan tiap unit jam ke 2:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P1 \leq 600$$

$$100 \leq P2 \leq 400$$

$$50 \leq P3 \leq 200$$

Syarat *ramp-rate*:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1)$$

Sehingga didapatkan batasan unit baru dengan memperhatikan kedua batasan.
Batasan baru tiap unit jam ke 2:

$$425.20 \leq P1 \leq 445.20$$

$$269.97 \leq P2 \leq 329.97$$

$$110.66 \leq P3 \leq 150.66$$

Besarnya nilai rugi-rugi daya pada iterasi 1:

$$Ploss = 0.00003P1^2 + 0.00009P2^2 + 0.00012P3^2$$
$$P1 = \frac{425.2 + 445.2}{2} = 435.2$$

$$P2 = \frac{269.97 + 329.97}{2} = 299.97$$

$$P3 = \frac{110.66 + 150.66}{2} = 130.66$$

$$Ploss = 15.8$$

Menentukan besar beban baru akibat penambahan nilai rugi-rugi pada iterasi 1,

$$Ptarget = Ploss + Pload$$

$$Ptarget = 15.8 + 800 = 815.8$$

Menetukan lambda *minimum* dan lambda *maximum*:

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(435.2) = 0.02611$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(299.97) = 0.05399$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(130.66) = 0.03136$$

$$Pfi = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1 - 0.02611}$$

$$Pf2 = \frac{1}{1 - 0.05399}$$

$$Pf3 = \frac{1}{1 - 0.03136}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 9.3293$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.7274$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.5284$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.1990$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 1,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 815.8 MW.

$$\lambda_{iter1} = 9.5284 \ldots\ldots\ldots totalgen = 865.83$$

$$\lambda_{iter2} = 9.3293 \ldots\ldots\ldots totalgen = 805.824$$

$$\lambda_{iter3} = 9.4288 \ldots\ldots\ldots totalgen = 821.56$$

$$\lambda_{iter4} = 9.3791 \ldots\ldots\ldots totalgen = 810.83$$

$$\vdots$$

$$\lambda_{iter13} = 9.4122 \ldots\ldots\ldots totalgen = 815.8$$

$$P1 = \frac{\lambda(1 - 0.02611) - 7.92}{2 \times 0.001562} = \frac{9.4122(1 - 0.02611) - 7.92}{2 \times 0.001562}$$
$$425.2 \leq P1 \leq 445.2$$
$$P1 = 425.2$$

$$P2 = \frac{\lambda(1 - 0.05399) - 7.85}{2 \times 0.00194} = \frac{9.4122(1 - 0.05399) - 7.85}{2 \times 0.00194}$$
$$269.97 \leq P2 \leq 329.97$$
$$P2 = 271.6$$

$$P3 = \frac{\lambda(1 - 0.03136) - 7.97}{2 \times 0.00482} = \frac{9.4122(1 - 0.03136) - 7.97}{2 \times 0.00482}$$
$$110.66 \leq P3 \leq 150.66$$
$$P3 = 118.9$$

Nilai $P1, P2, P3$ yang didapatkan kemudian digunakan untuk mencari rugi-rugi iterasi 2

$$PLoss = 0.00003(425.2^2) + 0.00009(271.6^2) + 0.00012(118.9^2)$$

$$Ploss = 13.8$$

Menentukan besar beban baru akibat penambahan nilia rugi-rugi pada iterasi 2,

$$Ptarget = Ploss + Pload$$

$$Ptarget = 13.8 + 800 = 813.8$$

Menetukan lambda *minimum* dan lambda *maximum*:

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(425.2) = 0.025512$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(271.6) = 0.048888$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(118.9) = 0.028536$$

$$Pfi = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1 - 0.025512}$$

$$Pf2 = \frac{1}{1 - 0.048888}$$

$$Pf3 = \frac{1}{1 - 0.028536}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 9.3024$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.6993$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.5009$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.1985$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 2,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 813.8 MW.

$$\lambda_{iter1} = 9.5009 \dots \dots \dots totalgen = 864.87$$

$$\lambda_{iter2} = 9.3024 \dots \dots \dots totalgen = 805.82$$

$$\lambda_{iter3} = 9.4016 \dots \dots \dots totalgen = 827.28$$

$$\lambda_{iter4} = 9.3520 \dots \dots \dots totalgen = 810.82$$

$$\vdots$$

$$\lambda_{iter13} = 9.3626 \dots \dots \dots totalgen = 813.8$$

$$P1 = \frac{\lambda(1 - 0.0255) - 7.92}{2 \times 0.001562} = \frac{9.3626(1 - 0.0255) - 7.92}{2 \times 0.001562}$$
$$425.2 \leq P1 \leq 435.2$$
$$P1 = 425.2$$

$$P2 = \frac{\lambda(1 - 0.0489) - 7.85}{2 \times 0.00194} = \frac{9.3626(1 - 0.0489) - 7.85}{2 \times 0.00194}$$
$$269.97 \leq P2 \leq 329.97$$
$$P2 = 271.88$$

$$P3 = \frac{\lambda(1 - 0.02854) - 7.97}{2 \times 0.00482} = \frac{9.3626(1 - 0.02854) - 7.97}{2 \times 0.00482}$$
$$110.66 \leq P3 \leq 150.66$$
$$P3 = 116.75$$

Nilai $P1, P2, P3$ yang didapatkan pada iterasi ke 2, kemudian digunkan untuk mencari rugi-rugi iterasi ke 3. Hal ini kan terus berlangsung hingga didapatkan nilai $P1, P2, P3$ pada rugi-rugi iterasi ke 11. Nilai rugi-rugi yang didapatkan pada iterasi ke 11 adalah sebesar 13.7 Sehingga total load yang diinginkan akan sebesar 813.7.

Didapatkan nilai lambda yang telah diiterasi untuk rugi-rugi pada iterasi ke 11 sebesar 9.3608, sehingga nilai yang pembangkitan setiap unit yang didapatkan sebesar,

$$P1 = 425.2$$

$$P2 = 271.5$$

$$P3 = 117.0$$

Sehingga nilai biaya pada beban jam ke 2 didapatkan

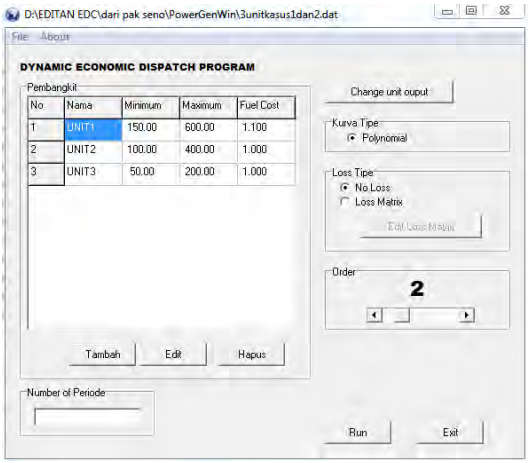
$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F1 = 561 + 7.92(425.2) + 0.001562(425.2^2)$$
$$F1 = 4210.95 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$
$$F2 = 301 + 7.85(271.5) + 0.00194(271.5^2)$$
$$F2 = 2584.18 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(117) + 0.00482(117^2)$$
$$F3 = 1076.54 \text{ (\$/jam)}$$

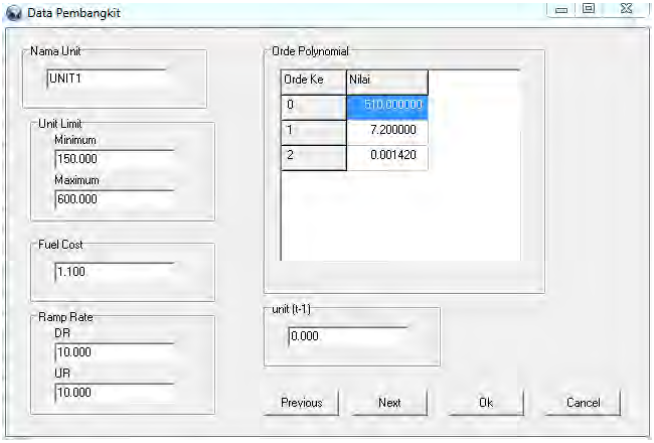
## 3.5 Aplikasi Perhitungan DED Iterasi Lambda

Berikut akan dijelaskan mengenai aplikasi perhitungan yang digunkan dalam Tugas Akhir ini. Seperti yang telah dinyatakan sebelumnya aplikasi perhitungan dibuat dengan menggunkan Delphi. Aplikasi yang digunakan khusus untuk melakukan perhitungan kasus penyelesaian DED. Berikut adalah tampilan aplikasi pada Gambar 3.3.




**Gambar 3.3** Tampilan utama aplikasi perhitugan DED

Tombol ‘Tambah’, ‘Edit’, dan ‘Hapus’ digunkan untuk mengisi data karakteristik pembagit yang telah ditentukan. Label kosong pada ‘Number of Periode’ diisi untuk menentukan jumlah periode beban yang ingin diperhitungkan DED. Ketika tombol ‘Edit’ digunkan maka akan muncul form pengisian data pembangkit seperti yang dipelihatkan pada Gambar 3.4.

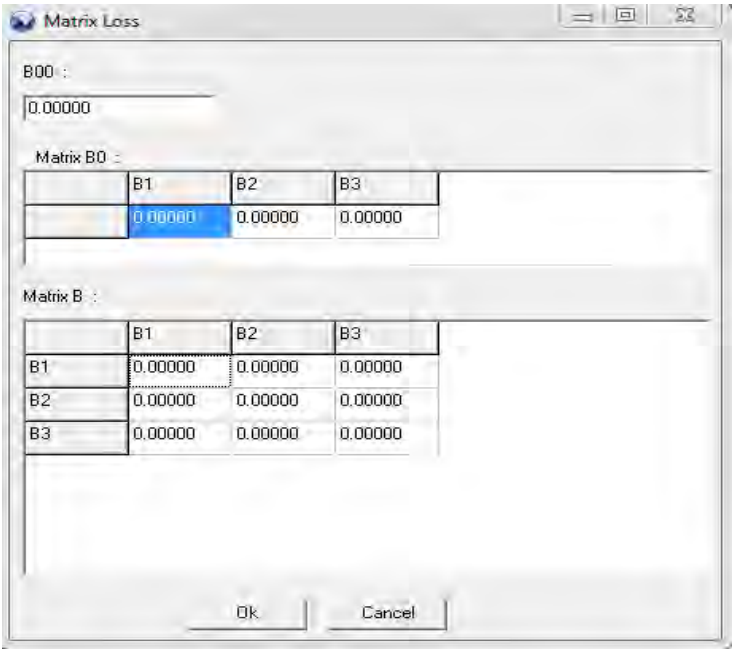



**Gambar 3.4** Tampilan pengisian karakteristik pembangkit

Gambar 3.4 memperlihatkan, bagian label ‘Minimum’ dan ‘Maximum, diisi dengan nilai batasan *minimum* dan *maximum* masing-masing unit. Label ‘Fuelcost’ diisi dengan nilai *fuelcost* setiap unit. Label ‘DR’ dan ‘UR’ berisikan nilai batsan *ramp-rate* setiap unit. Pada tabel ‘Orde Polynomial’ orde 0 berisi koefisen C, orde1 berisi koefisien B dan orde 2 berisi koefisien A pada persaman (2.13). Label ‘unit (t-1)’ diisikan nilai pembangkitan pada periode sebelumnya, jika diketahui.

Tombol ‘Next’ dan ‘Previous’ digunakan untuk memindah form pengisian data katarakteristik pembangkit sebelum dan sesudah. Tombol ‘Ok’ ditekan setelah data yang digunakan telah selesai diisi.

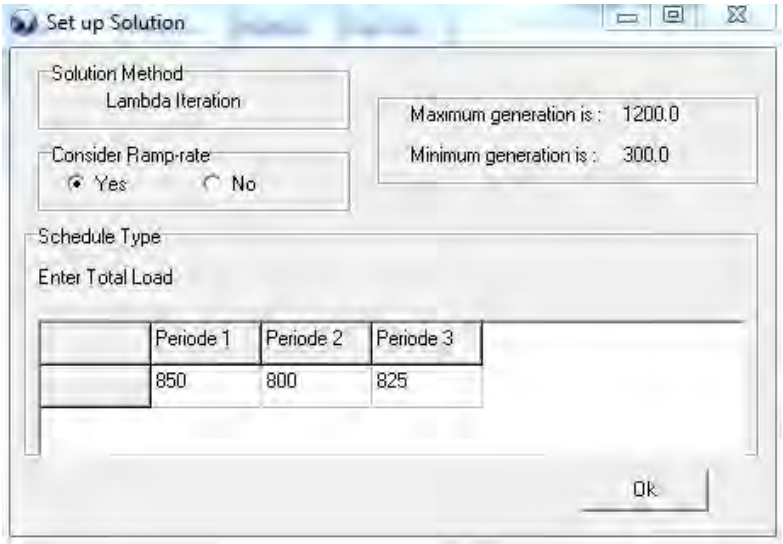
Pada Gambar 3.3, diberikan dua pilihan pengerjaan yaitu, ketika tidak melihat nilai rugi-rugi dengan memilih pilihan ‘No loss’, atau dengan memperhatikan nilia rugi-rugi dengan memilih ‘Loss Matrix’ yang kemudian dilanjutkan dengan menekan tombol ‘Edit Loss Marix’, maka akan keluar form pengisiaan data *Bloss matrix* seperti yang diperlihatkan pada Gambar 3.5,




**Gambar 3.5** Tampilan pengisian *Bloss matrix*

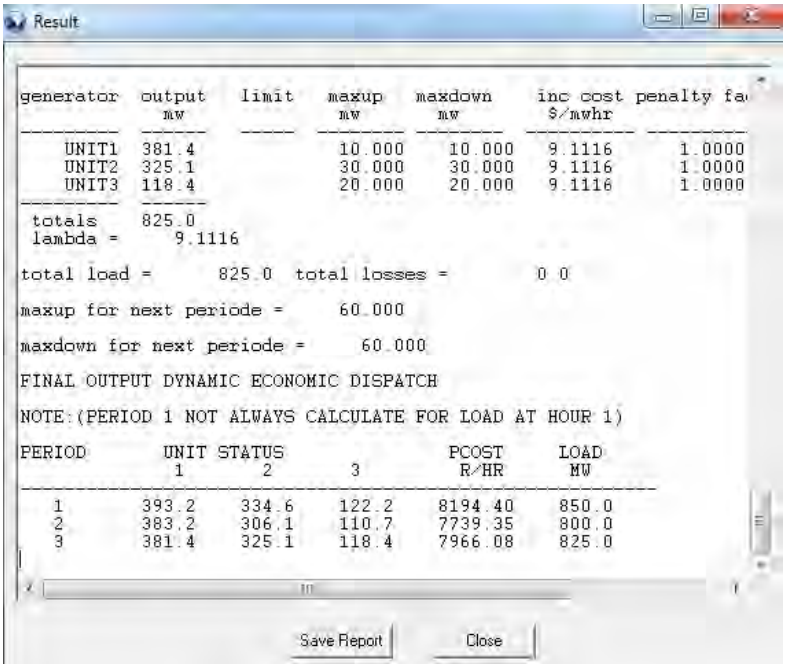
Pada Gambar 3.5 diberikan contoh ketika digunakan tiga unit pembangkitan. Sehingga kolom 'Matrix B0' akan berjumlah $1 \times 3$ dan 'Matrix B' akan berjumlah $3 \times 3$, sedangkan 'B00' akan selalu berisikan 1 kolom. Hal ini telah disesuikan dengan penggunaan rumus rugi-rugi yang konstan pada persamaan (2.27). Setelah semua bilangan yang dibutuhan telah terisi, maka tombol 'Ok' ditekan dan form akan kembali pada Gambar 3.3.

Setelah semua data yang digunakan telah diimputkan kemudian tekan tombol 'Run' pada Gambar 3.3, maka akan keluar form pengisian beban setiap periode yang diinginkan. Nilai beban yang digunakan adalah yang memenuhi batasan *ramp-rate* yang dimiliki setiap unit pembangkitan. Nilai beban juga tidak boleh melebihi kampasitas *maximum* pembangitan dan tidak boleh kurang dari kapasitas *minimum* dari pembangkitan. Form pengisian beban pada setiap periodenya dapat dilihat pada Gambar 3.6 berikut,




**Gambar 3.6** Tampilan pengisian beban

Pada Gambar 3.6 diberikan contoh ketika periode beban yang ingin dilakukan perhitungan DED sebanyak dua periode. Sehingga akan muncul kolom pengisian beban 'Periode 1' dan 'Periode 2'. Tombol 'Ok' ditekan setelah semua kolom pengisian beban disetiap periode telah disikan dan akan mucul form hasil keluaran perhitungan DED. Pada aplikasi ini diberikan pilihan pada 'Consider Ramp-rate', jika menetan pilihan 'Yes' maka DED akan diperhitungkan dengan melihat batasan *ramp-rate* yang telah diisi pada Gambar 3.4. Jika pilihan 'No' ditekan maka perhitungan DED akan dilakukan tanpa memperhatikan batasan *ramp-rate* yang tentu saja sangat memungkinkan hasil perhitungan nantinya akan melanggar batasan *ramp-rate*. Hasil perhitungan akan ditampilkan seperti yang ditunnukkan pada Gambar 3.7.




**Gambar 3.7** Tampilan hasil perhitungan DED

Pada Gambar 3.7 digunakan contoh perhitungan DED 3 unit generator pada 2 periode beban. Tombol 'Save Report' berfungsi untuk menyimpan hasil perhitungan DED yang telah dilakukan. Data yang disimpan akan berupa *file.txt*.

# BAB 4
# ANALISA DAN SIMULASI DATA

Pada Bab ini akan menampilkan hasil simulasi perhitungan *Dynamic Economic Dispatch* dengan menggunkan iterasi lambda. Analisa yang dilakukan adalah melihat pengaruh *ramp-rate* dalam perhitungan DED. Diberikan beberap contoh kasus pada Bab 4 ini, yaitu kasus 1, kasus 2 sebagai uji validasi. Sedangkan kasus 3, kasus 4, kasus 5, dan kasus 6 digunakan untuk melihat pengaruh perhitungan DED dengan memperhatikan batasan *ramp-rate.* Pada kasus 5 dan kasus 6 akan terlihat pengaruh *ramp-rate* yang mengakibatkan selektivitas pembangkitan setiap unit.

## 4.1 Kasus 1

Pada kasus 1 digunakan data pembangkitan melihat dari reverensi [3] pada contoh kasus 3A, dan data beban dilanjutkan pada interval waktu 3 jam kedepan. Diberikan parameter *up-rate* (UR) dan *down_rate* (DR) sebagai batasan ramp-rate. Kasus ini akan melihat tiga periode beban dengan interval waktu satu jam setiap periodenya. Tujuan dari kasus 1 adalah sebagai uji validasi perhitungan DED telah tidak melebihi batasan yang diberikan ketika memperhatikan batasan *ram-rate.* Berikut adalah data yang digunakan dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

**Tabel 4.1** Data karakteristik pembangkit 3 unit

| | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| A | 0.00142 | 0.00194 | 0.0048 |
| B | 7.2 | 7.85 | 7.97 |
| C | 510 | 310 | 78 |
| Pmin | 150 | 100 | 50 |
| Pmax | 600 | 400 | 200 |
| UR | 10 | 30 | 20 |
| DR | 10 | 30 | 20 |
| Fuelcost | 1.1 | 1 | 1 |

**Tabel 4.2** Data beban pembangkit 3 unit

| Jam | 1 | 2 | 3 |
|---|---|---|---|
| Beban | 850 | 800 | 825 |

Setelah dilakukan pengolahan data dengan menggunakan aplikasi perhitungan Delphi yang telah dibuat, maka didapatkan hasil pembangkitan setiap unit pada Tabel 4.3 sebagai berikut:

**Tabel 4.3** Pembangkitan kasus 1 dengan *ramp-rate*

| Jam | Daya Pembangkit (MW) | | |
|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 |
| 1 | 393.2 | 334.6 | 122.2 |
| 2 | 383.2 | 306.1 | 110.7 |
| 3 | 381.4 | 325.1 | 118.4 |

Dari hasil pembangkitan dapat dilihat perubahan setiap unit untuk mencapai beban pada periode berikutnya. Perubahan dapat dilihat pada Tabel 4.4 berikut,

**Tabel 4.4** Perubahan pembangkit/jam kasus 1 dengan *ramp-rate*

| Unit | Jam | |
|---|---|---|
| | 1 ke 2 | 2 ke 3 |
| Unit 1 | -10 | -1.8 |
| Unit 2 | -28.5 | 19 |
| Unit 3 | -11.5 | 7.7 |

Tabel 4.4 menyatakan bahwa perubahan setiap unit pada interval setiap satu jam tidak melebihi batasan *ramp-rate*. Ini berarti aplikasi perhitungan Delphi dapat dijalankan sesuai dengan batasan yang telah diberikan.

Analisa pada kasus 1 dilanjutkan dengan menjalankan DED dengan tidak memperhatikan batasan *ramp-rate*. Didapatkan hasil pembangkitan setiap unit yang berbeda dengan DED yang memperhatikan batasan *ramp-rate*. Hal ini bertujuan untuk melihat hasil keluaran pembangkitan setiap unit pada setiap periode dapat dilakukan DED tanpa melihat

batasan *ramp-rate* akan memiliki pembangkitan yang sama dengan hasil yang ditunjukkan pad Tabel 4.4. Hasil pembangkitan dapat dilihat pada Tabel 4.5 berikut,

**Tabel 4.5** Pembangkitan kasus 1 tanpa *ramp-rate*

| Jam | Daya Pembangkit (MW) | | |
|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 |
| 1 | 393.2 | 334.6 | 122.2 |
| 2 | 369.7 | 315.7 | 114.6 |
| 3 | 381.4 | 325.2 | 118.4 |

Akan tetapi perubahan setiap unit untuk beban setiap jam tidak memenuhi batasan *ramp-rate*, meskipun daya yang dibangkitkan memenuhi batasan *maximum* dan *minimum* pembangkitan setiap unit. Perubahan pembangkitan dapat dilihat pada Tabel 4.6 berikut,

**Tabel 4.6** Perubahan pembangkit/jam kasus 1 tanpa *ramp-rate*

| Unit | Jam | |
|---|---|---|
| | 1 ke 2 | 2 ke 3 |
| Unit 1 | -23.5 | 11.7 |
| Unit 2 | -18.9 | 9.5 |
| Unit 3 | -7.6 | 3.8 |

## 4.2 Kasus 2

Kasus 1 kemudian dilanjutkan dengan melihat *Bloss matrix* yang telah diberikan dan diperhitungkan dengan melihat batasan *ramp-rate*. Pada kasus 2 ini bertujuann untuk melihat hasil perhitungan DED dengan memperhatikan *ramp-rate* telah benar jika diperhitungkan dengan rugi-rugi yang didapatkan dari *Bloss matrix*, dimana nilai pembangkitan diharuskan lebih besar daripada tidak memperhitungkan rugi-rugi.

**Tabel 4.7** *Bloss matrix* kasus 1

| Bij | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.00003 | 0 | 0 |
| 2 | 0 | 0.00009 | 0 |
| 3 | 0 | 0 | 0.00012 |

Dari penjalana perhitungan dengan aplikasi perhitungan DED yang telah dibuat didapatkan hasil yang dapat dilihat dalam Tabel 4.8 berikut,

**Tabel 4.8** Pembangkitan kasus 1 dengan rugi-rugi

| Jam | Daya Pembangkit (MW) | | |
|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 |
| 1 | 435.2 | 300 | 130.7 |
| 2 | 425.2 | 271.5 | 117 |
| 3 | 421.9 | 291.4 | 126.6 |

Didapatkan hasil pembangkitan setiap unit yang lebih besar daripada pembangkitan unit yang tidak memperhitungkan nilai rugi-rugi pada Tabel 4.3, hal ini dikarenakan rugi-rugi daya menjadi beban tambahan pada penjadwalan pembangkitan.

## 4.3 Kasus 3

Pada kasus 3 data yang digunakan adalah 5 unit pembangkit [8]. DED diperhitngkan dengan data beban yang berubah selama 24 jam. Berikut adalah data karakteristik pembangkit,

**Tabel 4.9** Data karakteristik pembangkit 5 unit

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|---|
| A | 0.008 | 0.003 | 0.0012 | 0.001 | 0.0015 |
| B | 2 | 1.8 | 2.1 | 2 | 1.8 |
| C | 2.5 | 60 | 100 | 120 | 40 |
| Pmin | 10 | 20 | 30 | 40 | 50 |
| Pmax | 75 | 125 | 175 | 250 | 300 |

**Tabel 4.9** Data karakteristik pembangkit 5 unit (lanjutan)

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|---|
| DR | 30 | 30 | 40 | 50 | 50 |
| UR | 30 | 30 | 40 | 50 | 50 |

**Tabel 4.10** Data beban pembangkit 5 unit

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 410 | 9 | 690 | 17 | 558 |
| 2 | 435 | 10 | 704 | 18 | 608 |
| 3 | 475 | 11 | 720 | 19 | 654 |
| 4 | 530 | 12 | 740 | 20 | 704 |
| 5 | 558 | 13 | 704 | 21 | 680 |
| 6 | 608 | 14 | 690 | 22 | 605 |
| 7 | 626 | 15 | 654 | 23 | 527 |
| 8 | 654 | 16 | 580 | 24 | 463 |

Perhitungan dilakukan dua kali yaitu, *Dynamic Economic Dispatch* ketika tidak memperhatikan batasan *ramp-rate* dan *Dynamic Economic Dispatch* dengan memperhatikan batasan *ramp-rate*. Hal ini bertujuan untuk melihat hasil perhitungan dengan adanya efek batasan *ramp-rate*

**Tabel 4.11** Pembangkitan kasus 3 tanpa *ramp-rate*

| Jam | Daya Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 1 | 15.3 | 72.6 | 61.6 | 118.5 | 145.6 |
| 2 | 16.4 | 75.4 | 68.8 | 127 | 151.4 |
| 3 | 18.2 | 80 | 80.4 | 140.6 | 160.7 |
| 4 | 20.6 | 86.3 | 96.4 | 159.3 | 173.5 |
| 5 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |
| 6 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 7 | 24.8 | 97.3 | 124.4 | 192 | 195.8 |

**Tabel 4.11** Pembangkitan kasus 3 tanpa *ramp-rate* (lanjutan)

| Jam | Data Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 8 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 9 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 10 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 11 | 29 | 108.1 | 151.9 | 224.2 | 217.8 |
| 12 | 29.9 | 110.4 | 157.7 | 231 | 222.5 |
| 13 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 14 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 15 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 16 | 22.8 | 92 | 110.9 | 176.3 | 185.1 |
| 17 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |
| 18 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 19 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 20 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 21 | 27.2 | 103.5 | 140.1 | 210.5 | 208.5 |
| 22 | 23.9 | 94.9 | 118.2 | 184.9 | 190.9 |
| 23 | 20.5 | 85.9 | 95.5 | 158.3 | 172.8 |
| 24 | 17.6 | 78.6 | 76.9 | 136.5 | 157.9 |

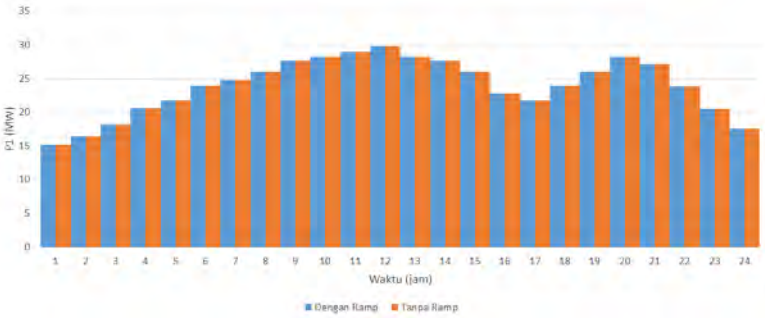
**Tabel 4.12** Pembangkitan kasus 3 dengan *ramp-rate*

| Jam | Daya Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 1 | 15.3 | 72.6 | 61.6 | 118.5 | 145.6 |
| 2 | 16.4 | 75.4 | 68.8 | 127 | 151.4 |
| 3 | 18.2 | 80 | 80.4 | 140.6 | 160.7 |
| 4 | 20.6 | 86.3 | 96.4 | 159.3 | 173.5 |
| 5 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |

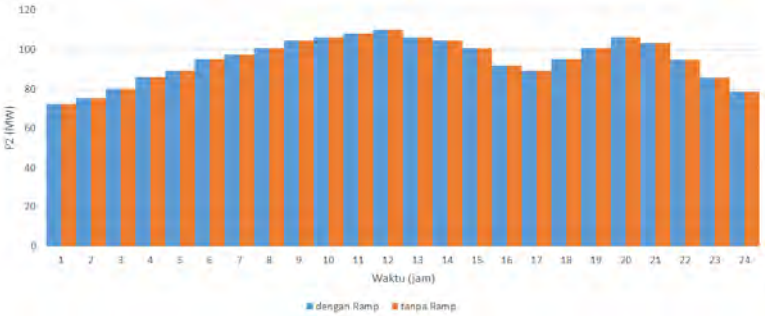
**Tabel 4.12** Pembangkitan kasus 3 dengan *ramp-rate* (lanjutan)

| Jam | Data Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 6 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 7 | 24.8 | 97.3 | 124.4 | 192 | 195.8 |
| 8 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 9 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 10 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 11 | 29 | 108.1 | 151.9 | 224.2 | 217.8 |
| 12 | 29.9 | 110.4 | 157.7 | 231 | 222.5 |
| 13 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 14 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 15 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 16 | 22.8 | 92 | 110.9 | 176.3 | 185.1 |
| 17 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |
| 18 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 19 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 20 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 21 | 27.2 | 103.5 | 140.1 | 210.5 | 208.5 |
| 22 | 23.9 | 94.9 | 118.2 | 184.9 | 190.9 |
| 23 | 20.5 | 85.9 | 95.5 | 158.3 | 172.8 |
| 24 | 17.6 | 78.6 | 76.9 | 136.5 | 157.9 |

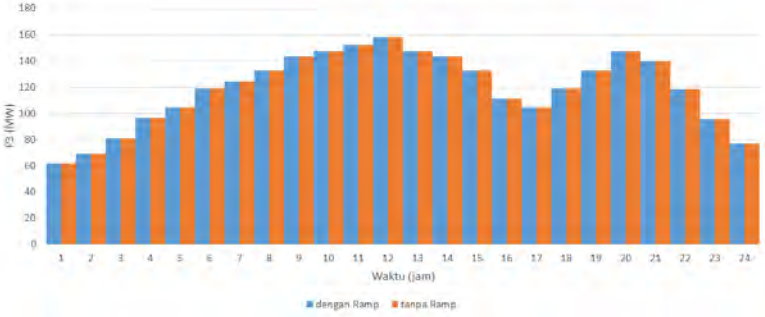
Dari data yang didapatkan dari Tabel 4.11 dan Tabel 4.12 dibuat garafik perubahan beban setiap unit. Grafik setiap unit dilihat perubahan kenaikan dan kenaikan pada setiap jamnya. Perubahan dapat dilihat pada Gambar 4.1 hingga Gambar 4.5.



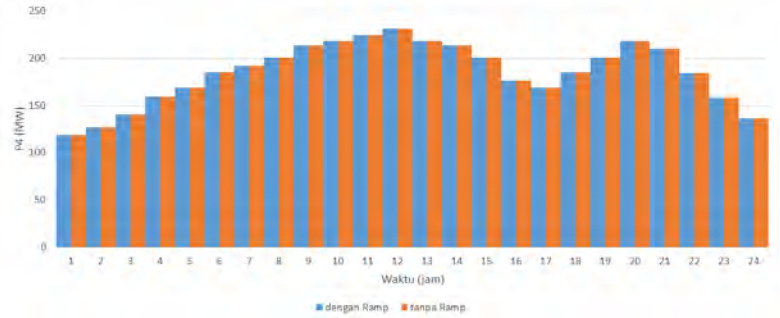

**Gambar 4.1** Grafik pembangkitan unit 1 kasus 3



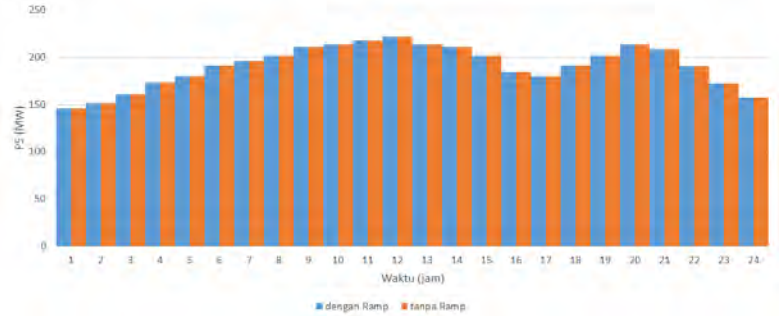

**Gambar 4.2** Grafik pembangkitan unit 2 kasus 3




**Gambar 4.3** Grafik pembangkitan unit 3 kasus 3




**Gambar 4.4** Grafik pembangkitan unit 4 kasus 3




**Gambar 4.5** Grafik pembangkitan unit 5 kasus 3

**Tabel 4.13** Perbandingan total biaya kasus 3

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 40121.08 |
| Dengan *Ramp-rate* | 40121.08 |

Dari hasil data pembangkitan yang didapatkan, terlihat tidak adanya perubahan pembangkitan setiap unit ketika tidak memperhatikan *ramp-rate* dan dengan memperhatikan *ramp-rate*. Hal ini dikarenakan nilai optimal dari unit yang memperhatikan *ramp-rate* masih berada didalam batasan pembangkitan *maximum* dan *minimum* generator, sehingga tidak terjadi seleksi penjadwalan pembangkit dan total biaya yang dihasilkan sama. Hal ini dapat dilihat pada Tabel 4.13.

## 4.4 Kasus 4

Kasus 4 menggunakan data karakteristik 6 pembangkit [9]. Sama dengan kasus 3, pada kasus 4 akan dilakukan perhitunga sebanyak dua kali yaitu ketika tidak memperhatikan batasan *ramp-rate*, dan perhitungan ketika memperhatikan batasan *ramp-rate*. Data yang digunakan adalah sebagai berikut,

**Tabel 4.14** Data karakteristik pembangkit 6 unit

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 |
|---|---|---|---|---|---|---|
| A | 0.007 | 0.0095 | 0.009 | 0.009 | 0.008 | 0.0075 |
| B | 7 | 10 | 8 | 11 | 10.5 | 12 |
| C | 240 | 200 | 220 | 200 | 220 | 190 |
| Pmin | 100 | 50 | 80 | 50 | 50 | 50 |
| Pmax | 500 | 200 | 300 | 150 | 200 | 120 |
| DR | 120 | 90 | 100 | 90 | 90 | 90 |
| UR | 80 | 50 | 65 | 50 | 50 | 50 |

**Tabel 4.15** Data beban pembangkitan 6 unit

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 955 | 9 | 1126 | 17 | 1221 |
| 2 | 942 | 10 | 1150 | 18 | 1202 |
| 3 | 953 | 11 | 1201 | 19 | 1159 |
| 4 | 930 | 12 | 1235 | 20 | 1092 |
| 5 | 935 | 13 | 1190 | 21 | 1023 |
| 6 | 963 | 14 | 1251 | 22 | 984 |
| 7 | 989 | 15 | 1263 | 23 | 975 |
| 8 | 1023 | 16 | 1250 | 24 | 960 |

Dari hasil perhitungan data Tabel 4.14 dan data Tabel 4.15 didapatkan hasil pembangkitan sebagai berikut.

**Tabel 4.16** Pembangkitan kasus 4 tanpa *ramp-rate*

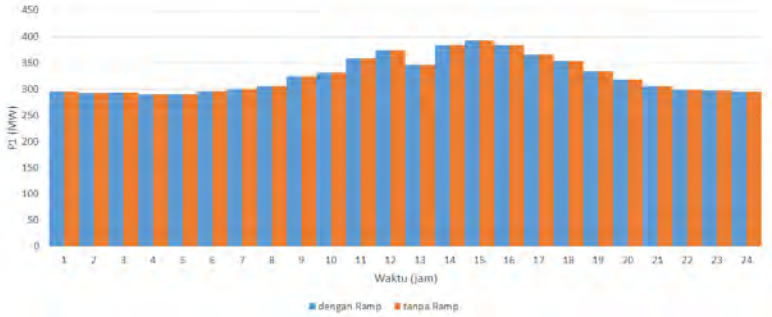
| Jam | Daya Pembangkit (MW) | | | | | |
|---|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 |
| 1 | 294.2 | 95.9 | 260.5 | 150 | 149.8 | 51.4 |
| 2 | 291.6 | 93.6 | 257 | 150 | 145.7 | 50 |
| 3 | 293.9 | 95.6 | 260 | 150 | 149.2 | 50.9 |
| 4 | 288.8 | 91.1 | 253.5 | 150 | 141.6 | 50 |
| 5 | 290 | 92.1 | 255 | 150 | 143.3 | 50 |
| 6 | 295.6 | 97.2 | 262.5 | 150 | 151.9 | 53.2 |
| 7 | 300 | 101.2 | 268.9 | 150 | 159 | 59.1 |
| 8 | 305.8 | 106.5 | 277.3 | 150 | 168.2 | 66.9 |
| 9 | 324 | 123.2 | 300 | 150 | 197.5 | 91.2 |
| 10 | 331.7 | 130.3 | 300 | 150 | 200 | 100.9 |
| 11 | 358.1 | 154.6 | 300 | 150 | 200 | 120 |
| 12 | 374 | 169.2 | 300 | 150 | 200 | 120 |
| 13 | 346 | 143.6 | 300 | 150 | 200 | 119 |
| 14 | 384.3 | 178.7 | 300 | 150 | 200 | 120 |
| 15 | 392.1 | 185.9 | 300 | 150 | 200 | 120 |
| 16 | 383.7 | 178.1 | 300 | 150 | 200 | 120 |
| 17 | 365.1 | 161 | 300 | 150 | 200 | 120 |
| 18 | 353 | 149.9 | 300 | 150 | 200 | 120 |
| 19 | 334 | 133.2 | 300 | 150 | 200 | 104.9 |
| 20 | 317.5 | 117.2 | 294.4 | 150 | 187.2 | 82.7 |
| 21 | 305.8 | 106.5 | 277.3 | 150 | 168.3 | 66.9 |
| 22 | 299.2 | 100.4 | 267.6 | 150 | 157.6 | 58 |
| 23 | 297.7 | 99 | 265.4 | 150 | 155.2 | 55.9 |
| 24 | 295.1 | 96.7 | 261.8 | 150 | 151.1 | 52.5 |

**Tabel 4.17** Pembangkitan kasus 4 dengan *ramp-rate*

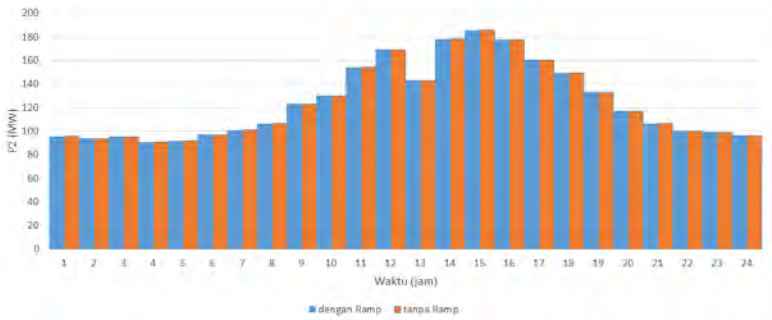
| Jam | Daya Pembangkit (MW) | | | | | |
|---|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 |
| 1 | 294.2 | 95.9 | 260.5 | 150 | 149.8 | 51.4 |
| 2 | 291.6 | 93.6 | 257 | 150 | 145.7 | 50 |
| 3 | 293.9 | 95.6 | 260 | 150 | 149.2 | 50.9 |
| 4 | 288.8 | 91.1 | 253.5 | 150 | 141.6 | 50 |
| 5 | 290 | 92.1 | 255 | 150 | 143.3 | 50 |
| 6 | 295.6 | 97.2 | 262.5 | 150 | 151.9 | 53.2 |
| 7 | 300 | 101.2 | 268.9 | 150 | 159 | 59.1 |
| 8 | 305.8 | 106.5 | 277.3 | 150 | 168.2 | 66.9 |
| 9 | 324 | 123.2 | 300 | 150 | 197.5 | 91.2 |
| 10 | 331.7 | 130.3 | 300 | 150 | 200 | 100.9 |
| 11 | 358.1 | 154.6 | 300 | 150 | 200 | 120 |
| 12 | 374 | 169.2 | 300 | 150 | 200 | 120 |
| 13 | 346 | 143.6 | 300 | 150 | 200 | 119 |
| 14 | 384.3 | 178.7 | 300 | 150 | 200 | 120 |
| 15 | 392.1 | 185.9 | 300 | 150 | 200 | 120 |
| 16 | 383.7 | 178.1 | 300 | 150 | 200 | 120 |
| 17 | 365.1 | 161 | 300 | 150 | 200 | 120 |
| 18 | 353 | 149.9 | 300 | 150 | 200 | 120 |
| 19 | 334 | 133.2 | 300 | 150 | 200 | 104.9 |
| 20 | 317.5 | 117.2 | 294.4 | 150 | 187.2 | 82.7 |
| 21 | 305.8 | 106.5 | 277.3 | 150 | 168.3 | 66.9 |
| 22 | 299.2 | 100.4 | 267.6 | 150 | 157.6 | 58 |
| 23 | 297.7 | 99 | 265.4 | 150 | 155.2 | 55.9 |
| 24 | 295.1 | 96.7 | 261.8 | 150 | 151.1 | 52.5 |

Didapatkan dari Tabel 4.16 dan Tabel 4.17 dibuat garafik perubahan beban setiap unit. Grafik setiap unit dilihat perubahan kenaikan dan penurunan pembangkitan pada setiap jamnya.

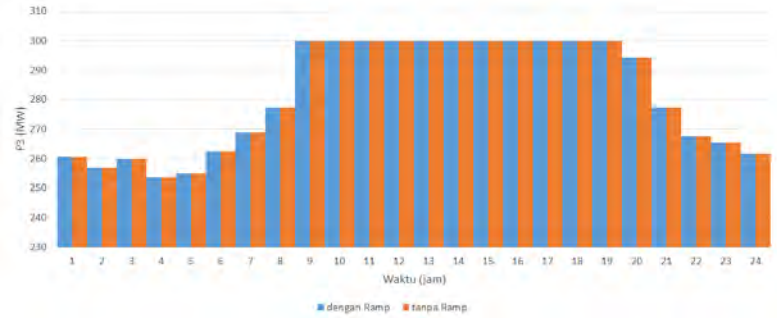
Pada gambar grafik didapatkan bentuk pola naik dan turunnya pembangkitan adalah sama ketika tidak memperhatikan *ramp-rate* dan ketika memperhatikan *ramp-rate*. Pola ini sama dengan contoh kasus 3. Untuk lebih jelasnya dapat dilihat pada Gambar 4.6 sampai 4.11 berikut.



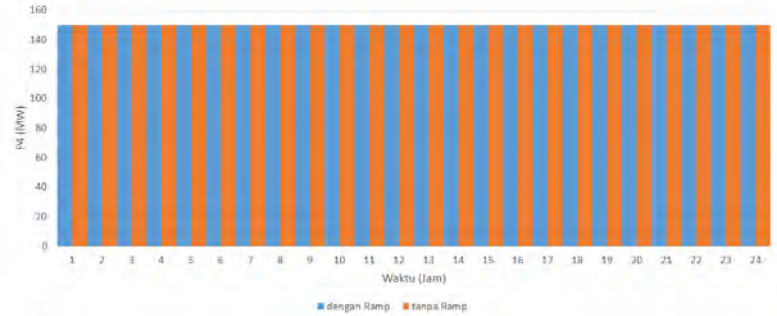

**Gambar 4.6** Grafik pembangkitan unit 1 kasus 4



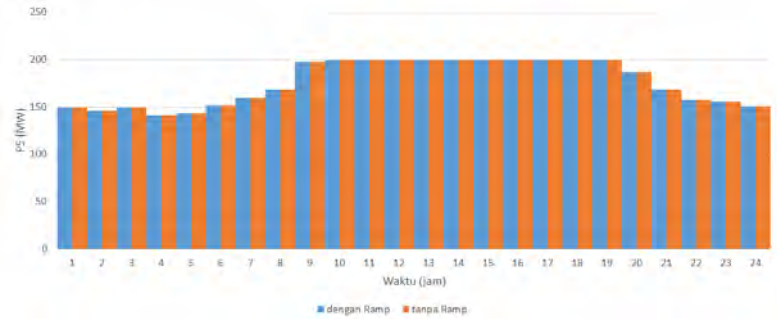

**Gambar 4.7** Grafik pembangkitan unit 2 kasus 4



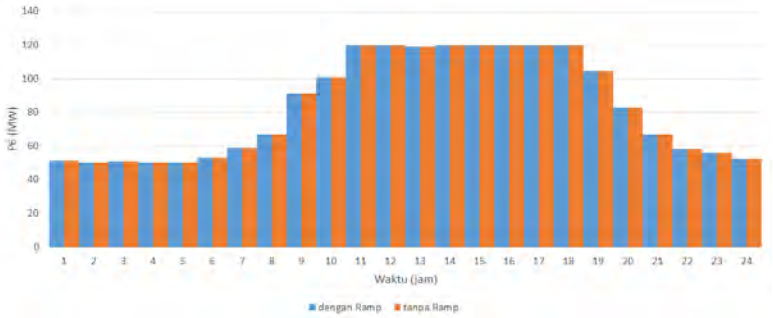

**Gambar 4.8** Grafik pembangkitan unit 3 kasus 4




**Gambar 4.9** Grafik pembangkitan unit 4 kasus 4




**Gambar 4.10** Grafik pembangkitan unit 5 kasus 4




**Gambar 4.11** Grafik pembangkitan unit 6 kasus 4

**Tabel 4.18** Perbandingan total biaya kasus 4

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 328724.3 |
| Dengan *Ramp-rate* | 328724.3 |

Dari data yang didapatkan menghasilkan pembangikitan yang sama ketika tidak memperhatikan *ramp-rate* dan ketika memperhatikan *ramp-rate*. Hal ini dikarenakan nilai optimal dari unit yang memperhatikan *ramp-rate* masih berada didalam batasan pembangkitan *maximum* dan *minimum* generator, sehingga tidak terjadi seleksi perubahan penjadwalan pembangkit dan total biaya pembangkitan yang dihasilkan sama. Dapat dilihat pada Tabel 4.18

## 4.5 Kasus 5

Digunakan data 10 unit [10] pembangkit. *Dynamic Economic Dispatch* diperhitung terhadap perubahan beban selama 24 jam. Pada kasus 5 juga melakukan dua kali perhitungan, yaitu DED tanpa memperhatikan *ramp-rate* dan DED dengan memperhatikan *ramp-rate*. Hal ini bertujuan untuk melihat pengaruh ramp-rate dalam perhitungan DED. Data karakteristik pembangkit yang digunakan adalah sebagai berikut,

**Tabel 4.19** Data karakteristik pembangkit 10 Unit

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|---|
| A | 0.1524 | 0.1058 | 0.028 | 0.0354 | 0.0211 |
| B | 38.5397 | 46.1591 | 40.3965 | 38.3055 | 36.3278 |
| C | 786.7988 | 451.3251 | 1049.9977 | 1243.5311 | 1658.5696 |
| Pmin | 150 | 135 | 73 | 60 | 73 |
| Pmax | 470 | 470 | 340 | 300 | 243 |
| DR | 80 | 80 | 80 | 50 | 50 |
| UR | 80 | 80 | 80 | 50 | 50 |
| | Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| A | 0.0179 | 0.0121 | 0.0121 | 0.109 | 0.1295 |
| B | 38.2704 | 36.5104 | 36.5104 | 39.5804 | 40.5407 |
| C | 1356.6592 | 1450.7045 | 1450.7045 | 1455.6056 | 1469.4026 |
| Pmin | 57 | 20 | 47 | 20 | 10 |
| Pmax | 160 | 130 | 120 | 80 | 55 |
| DR | 50 | 30 | 30 | 30 | 30 |
| UR | 50 | 30 | 30 | 30 | 30 |

**Tabel 4.20** Data beban pembangkitan 10 unit kasus 5

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 1036 | 9 | 1924 | 17 | 1480 |
| 2 | 1110 | 10 | 2022 | 18 | 1628 |
| 3 | 1258 | 11 | 2106 | 19 | 1776 |
| 4 | 1406 | 12 | 2150 | 20 | 1936 |
| 5 | 1480 | 13 | 2072 | 21 | 1924 |
| 6 | 1628 | 14 | 1924 | 22 | 1628 |
| 7 | 1702 | 15 | 1776 | 23 | 1332 |
| 8 | 1776 | 16 | 1554 | 24 | 1184 |

Dari perhitungan data Tabel 4.19 dan data Tabel 4.20 didapatkan

**Tabel 4.21** Pembangkitan kasus 5 tanpa *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |
| 3 | 73 | 79.6 | 126.9 | 192.3 | 224.9 | 293.8 |
| 4 | 74.8 | 92.5 | 129.9 | 181.6 | 207.5 | 261.9 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 20 | 24.2 | 36.3 | 53.1 | 61.5 | 79.2 |
| 10 | 11.8 | 16.7 | 26.9 | 41 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 218 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.6 | 278 | 335.9 | 385.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.6 | 218 | 157.4 | 150 | 150 | 150 |
| 2 | 365.4 | 278 | 190.6 | 135 | 135 | 135 |

**Tabel 4.21** Pembangkitan kasus 5 tanpa *ramp-rate* (lanjutan)

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 3 | 340 | 340 | 340 | 257.7 | 224.9 | 293.8 |
| 4 | 300 | 300 | 300 | 233.4 | 207.5 | 261.9 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 69.9 | 61.5 | 79.2 |
| 10 | 55 | 55 | 55 | 55 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 157.4 | 222.9 | 218 | 150 | 150 | 150 |
| 2 | 190.6 | 285.1 | 278 | 135 | 135 | 135 |
| 3 | 340 | 340 | 340 | 293.8 | 159.6 | 100.2 |
| 4 | 300 | 300 | 300 | 261.9 | 155.7 | 108.8 |
| 5 | 243 | 243 | 243 | 243 | 243 | 229.4 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 79.2 | 44.7 | 29.5 |
| 10 | 55 | 55 | 55 | 55 | 33.9 | 21.1 |

**Tabel 4.22** Pembangkitan kasus 5 dengan *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |

**Tabel 4.22** Pembangkitan kasus 5 dengan *ramp-rate* (lanjutan)

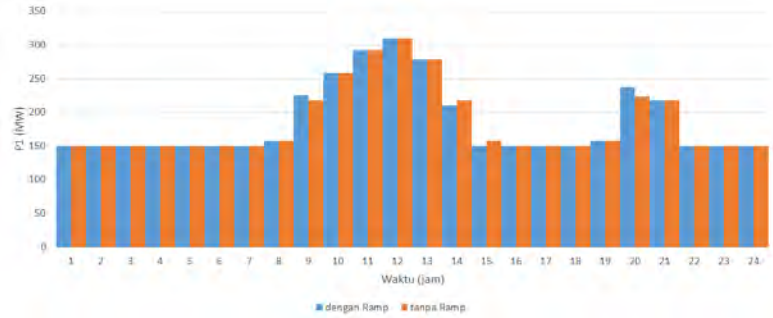
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 73 | 79.6 | 126.9 | 193.4 | 224.9 | 297.5 |
| 4 | 74.8 | 92.5 | 129.9 | 179.9 | 207.5 | 257.5 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 20 | 24.2 | 36.3 | 53.4 | 61.5 | 80 |
| 10 | 11.8 | 16.7 | 26.9 | 41.3 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 225.4 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.7 | 270.7 | 335.9 | 285.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.6 | 210.6 | 150 | 150 | 150 | 150 |
| 2 | 365.4 | 285.4 | 205.4 | 135 | 135 | 135 |
| 3 | 340 | 340 | 336.7 | 256.7 | 224.9 | 297.5 |
| 4 | 300 | 300 | 295.9 | 245.9 | 207.5 | 257.5 |

**Tabel 4.22** Pembangkitan kasus 5 dengan *ramp-rate* (lanjutan)

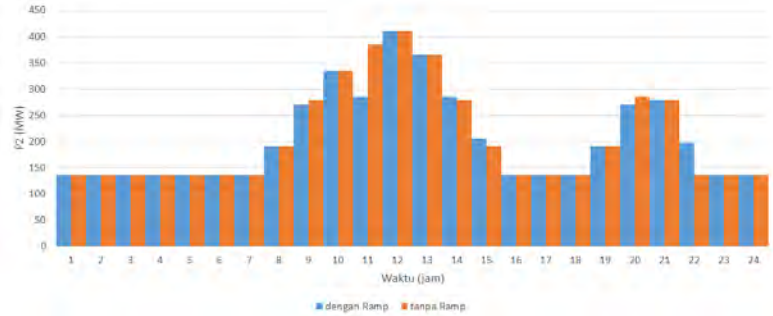
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 63.6 | 61.5 | 80 |
| 10 | 55 | 55 | 55 | 49.8 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 157.4 | 237.4 | 218 | 150 | 150 | 150 |
| 2 | 190.7 | 270.7 | 278 | 198 | 135 | 135 |
| 3 | 340 | 340 | 340 | 260 | 180 | 100 |
| 4 | 300 | 300 | 300 | 250 | 200 | 150 |
| 5 | 243 | 243 | 243 | 243 | 200 | 150 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 65.5 | 35.5 | 23.6 |
| 10 | 55 | 55 | 55 | 51.5 | 21.5 | 16.2 |

Didapatkan dari Tabel 4.21 dan Tabel 4.22 dibuat garafik perubahan beban setiap unit. Grafik setiap unit dilihat perubahan kenaikan dan penurunan pada setiap jamnya.

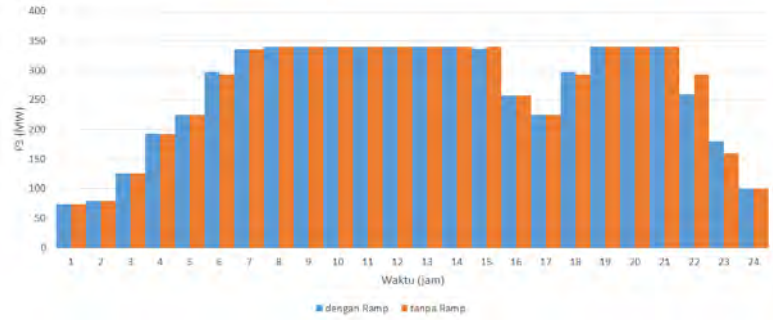
Pada gambar grafik didapatkan bentuk pola ketika tidak memperhatikan *ramp-rate* dapat naik dan turun dengan dengan melanggar batasan *ramp-rate*, sedangkan ketika memperhatikan *ramp-rate* pola naik dan turun pembangkit bertahap sesuai dengan nilai DR dan UR. Untuk lebih jelasnya dapat dilihat pada Gambar 4.12 sampai 4.16 berikut.



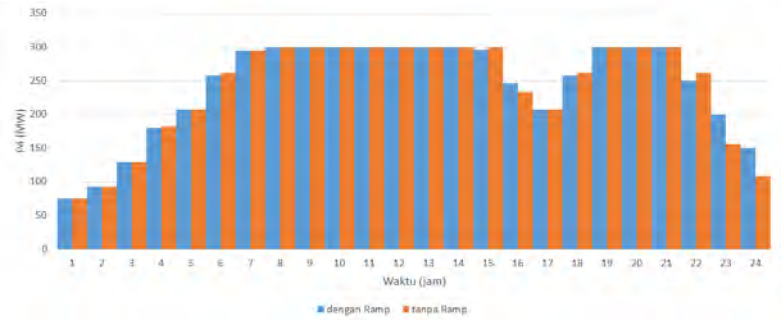

**Gambar 4.12** Grafik pembangkitan unit 1 kasus 5



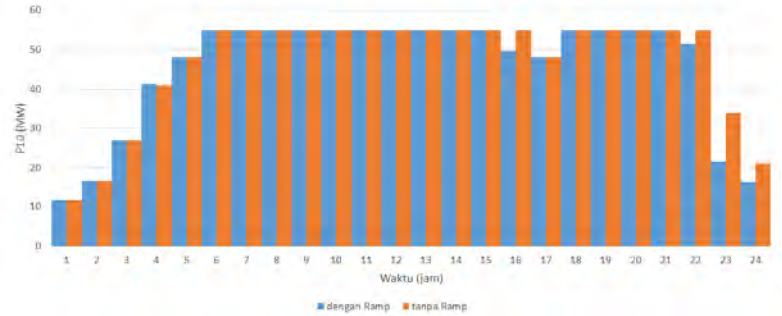

**Gambar 4.13** Grafik pembangkitan unit 2 kasus 5




**Gambar 4.14** Grafik pembangkitan unit 3 kasus 5




**Gambar 4.15** Grafik pembangkitan unit 4 kasus 5




**Gambar 4.16** Grafik pembangkitan unit 10 kasus 5

**Tabel 4.23** Perbandingan total biaya kasus 5

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 2298227 |
| Dengan *Ramp-rate* | 2300521 |

Dari hasil data yang didapatkan, terdapat perbedaan pembangkitan ketika tidak memperhitungkan *ramp-rate* dengan pembangkitan ketika memperhatikan *ramp-rate*. Hal ini dikarenakan batasan pemabangkitan yang melihat pengaruh *ramp-rate* akan semakin dipersempit dan seleksi penjadwalan pembangkitan akan semakin ketat.

Dengan adanya pengaruh ramp-rate pada kasus 5 ini didapatkan hasil total biaya yang lebih mahal $2294 daripada total biaya ketika tidak memperhatikan batasan *ramp-rate*.

## 4.6 Kasus 6

Pada contoh kasus 6 ini merupakan lanjutan dari kasus 5. Data karakteristik pembangkit yang digunakan sama dengan data yang ditampilkan pada Tabel 4.17. Sedangkan data beban yang digunakan pada kasus 6 dilakukan perubahan pada jam ke 19 dapat dilihat pada Tabel 4.24, dimana nantinya akan melihat efek perubahan penjadwalan pembangkit pada suatu periode dikarenakan *ramp-rate* nilai kombinasi *minimum* pembangkitan harus dilakukan perubahan untuk bisa memenuhi beban untuk interval waktu berikutnya.

**Tabel 4.24** Data beban pembangkit 10 unit kasus 6

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 1036 | 9 | 1924 | 17 | 1480 |
| 2 | 1110 | 10 | 2022 | 18 | 1628 |
| 3 | 1258 | 11 | 2106 | 19 | 1776 |
| 4 | 1406 | 12 | 2150 | 20 | 1972 |
| 5 | 1480 | 13 | 2072 | 21 | 1924 |
| 6 | 1628 | 14 | 1924 | 22 | 1628 |
| 7 | 1702 | 15 | 1776 | 23 | 1332 |
| 8 | 1776 | 16 | 1554 | 24 | 1184 |

Dari perhitungan data Tabel 4.19 dan data Tabel 4.24 didapatkan

**Tabel 4.25** Pembangkitan kasus 6 tanpa *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |
| 3 | 73 | 79.6 | 126.9 | 192.3 | 224.9 | 293.8 |
| 4 | 74.8 | 92.5 | 129.9 | 181.6 | 207.5 | 261.9 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |

**Tabel 4.25** Pembangkitan kasus 6 tanpa *ramp-rate* (lanjutan)

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 20 | 24.2 | 36.3 | 53.1 | 61.5 | 79.2 |
| 10 | 11.8 | 16.7 | 26.9 | 41 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 218 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.6 | 278 | 335.9 | 385.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.6 | 218 | 157.4 | 150 | 150 | 150 |
| 2 | 365.4 | 278 | 190.6 | 135 | 135 | 135 |
| 3 | 340 | 340 | 340 | 257.7 | 224.9 | 293.8 |
| 4 | 300 | 300 | 300 | 233.4 | 207.5 | 261.9 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |

**Tabel 4.25** Pembangkitan kasus 6 tanpa *ramp-rate* (lanjutan)

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 9 | 80 | 80 | 80 | 69.9 | 61.5 | 79.2 |
| 10 | 55 | 55 | 55 | 55 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 157.4 | 237.4 | 218 | 150 | 150 | 150 |
| 2 | 190.6 | 306.3 | 278 | 135 | 135 | 135 |
| 3 | 340 | 340 | 340 | 293.8 | 159.6 | 100.2 |
| 4 | 300 | 300 | 300 | 261.9 | 155.7 | 108.8 |
| 5 | 243 | 243 | 243 | 243 | 243 | 229.4 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 79.2 | 44.7 | 29.5 |
| 10 | 55 | 55 | 55 | 55 | 33.9 | 21.1 |

**Tabel 4.26** Pembangkitan kasus 6 dengan *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |
| 3 | 73 | 79.6 | 126.9 | 193.4 | 224.9 | 297.5 |
| 4 | 74.8 | 92.5 | 129.9 | 179.9 | 207.5 | 257.5 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |

**Tabel 4.26** Pembangkitan kasus 6 dengan *ramp-rate* (lanjutan)

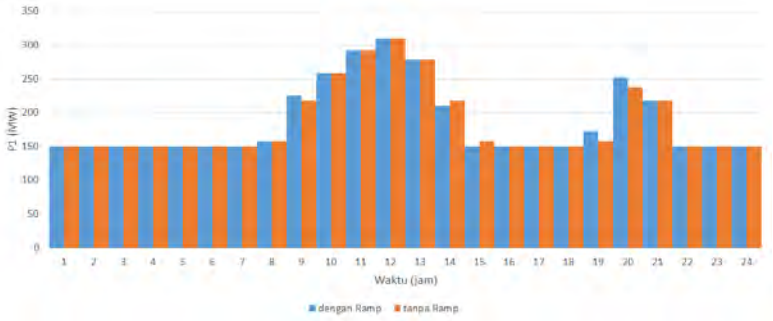
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 9 | 20 | 24.2 | 36.3 | 53.4 | 61.5 | 80 |
| 10 | 11.8 | 16.7 | 26.9 | 41.3 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 225.4 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.7 | 270.7 | 335.9 | 385.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.16 | 210.6 | 150 | 150 | 150 | 150 |
| 2 | 365.4 | 285.4 | 205.4 | 135 | 135 | 135 |
| 3 | 340 | 340 | 336.7 | 256.7 | 224.9 | 297.5 |
| 4 | 300 | 300 | 295.9 | 245.9 | 207.5 | 257.5 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 63.6 | 61.5 | 80 |
| 10 | 55 | 55 | 55 | 49.8 | 48.1 | 55 |

**Tabel 4.26** Pembangkitan kasus 6 dengan *ramp-rate* (lanjutan)

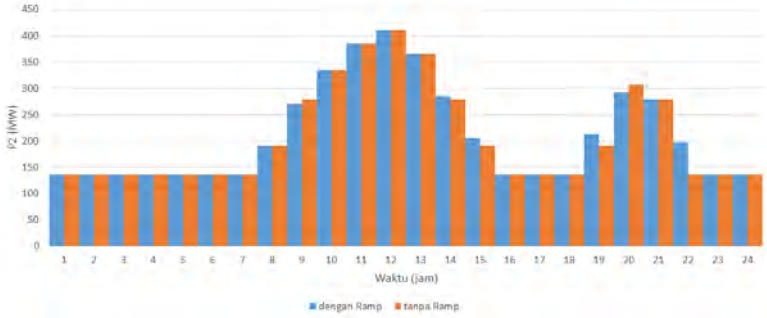
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 172.1 | 252.1 | 218 | 150 | 150 | 150 |
| 2 | 211.9 | 291.9 | 278 | 198 | 135 | 135 |
| 3 | 304 | 340 | 340 | 260 | 180 | 100 |
| 4 | 300 | 300 | 300 | 250 | 200 | 150 |
| 5 | 243 | 243 | 243 | 243 | 200 | 199.2 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 65.5 | 35.5 | 23.6 |
| 10 | 55 | 55 | 55 | 51.5 | 21.5 | 16.2 |

Dibuat grafik perubahan kenaikan dan penurunan pembangkitan dari data Tabel 4.25 dan Tabel 4.26.

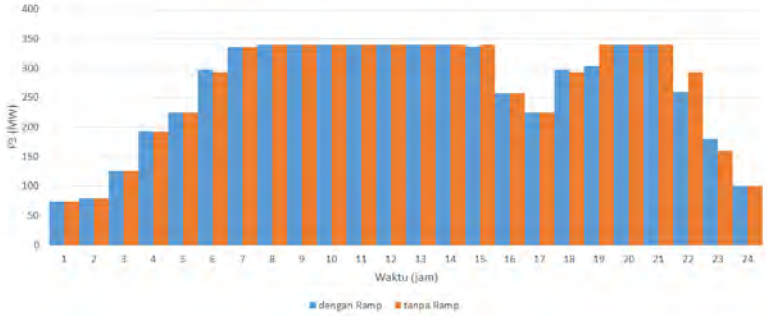
Pada gambar grafik didapatkan bentuk pola ketika tidak memperhatikan *ramp-rate* dapat naik dan turun dengan dengan melanggar batasan *ramp-rate*, sedangkan ketika memperhatikan *ramp-rate* pola naik dan turun pembangkit bertahap sesuai dengan nilai DR dan UR. Untuk lebih jelasnya dapat dilihat pada Gambar 4.17 sampai 4.21 berikut.



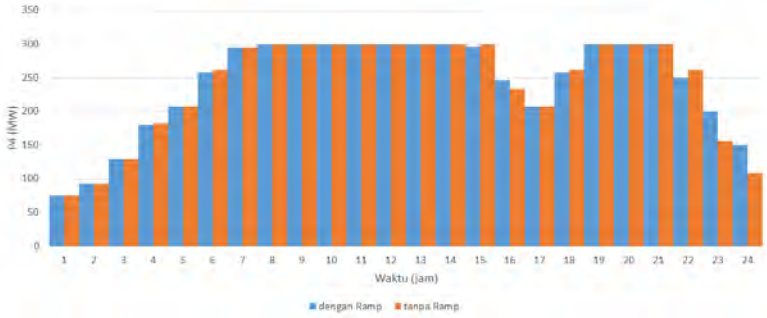

**Gambar 4.17** Grafik pembangkitan unit 1 kasus 6



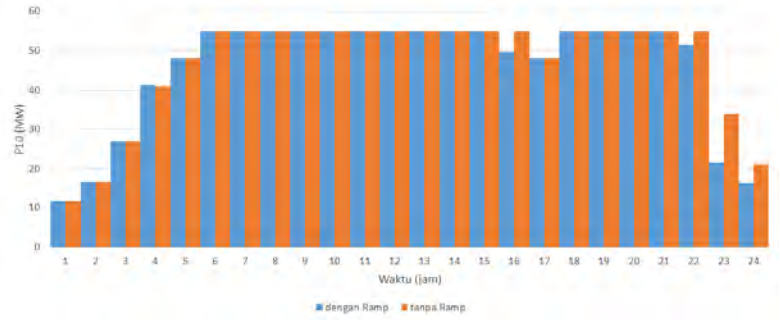

**Gambar 4.18** Grafik pembangkitan unit 2 kasus 6




**Gambar 4.19** Grafik pembangkitan unit 3 kasus 6




**Gambar 4.20** Grafik pembangkitan unit 4 kasus 6




**Gambar 4.21** Grafik pembangkitan unit 10 kasus 6

Pada kasus 6 ini terdapat kondisi dimana pada perhitungan DED beban jam ke 19, nilai kombinasi pembangkitan harus dirubah. Hali ini dikarenakan nilai pembangkitan yang didapatkan untuk jam ke 19 tidak dapat mengejar nilai pembangkitan pada jam ke 20 dikarenakan efek batasan *ramp-rate*. Untuk lebih jelasnya dapat dilihat pada tabel berikut,

**Tabel 4.27** Pembangkitan jam ke 19 sebelum dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 157.4 | 190.7 | 340 | 300 | 243 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 160 | 130 | 120 | 80 | 55 |

Sehingga dengan nilai pembangkitan yang didapatkan pada Tabel 4.27, maka akan didapatkan kemampuan *maximum* akibat *ramp-rate* tiap unit untuk naik pada jam ke 20 dapat dilihat pada Tabel 4.28

**Tabel 4.28** Nilai total *maximum* naik sebelum dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 80 | 80 | 0 | 0 | 0 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 0 | 0 | 0 | 0 | 0 |
| Total Kenaikan | | | | |
| 160 | | | | |

Untuk mencapai beban pada jam ke 20 sebesar 1972, dibutuhkkan kenaikan minimal sebesar 196. Jika nilai optimum pembangkitan pada jam ke 6 tidak dirubah maka tidak memungkinkan untuk bisa melanjutkan perhitungan DED pada jam ke 20, dikarenakan total kenaikan hanya sebesar 160. Oleh karena itu pembangkitan jam ke 6 dirubah menjadi Tabel 4.29 berikut.

**Tabel 4.29** Pembangkitan jam ke 19 setelah dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 172.1 | 211.9 | 304 | 300 | 243 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 160 | 130 | 120 | 80 | 55 |

Dari data Tabel 4.30 memungkinkan untuk melakukan perhitungan DED pada jam ke 20, dikarenakan total kenaikan sebesar 196 memenuhi

**Tabel 4.30** Nilai total *maximum* naik setelah dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 80 | 80 | 36 | 0 | 0 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 0 | 0 | 0 | 0 | 0 |
| Total Kenaikan | | | | |
| 196 | | | | |

**Tabel 4.31** Perbandingan total biaya kasus 6

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 2302142 |
| Dengan *Ramp-rate* | 2305528 |

Dari semua data yang ditunjukkan pada kasus 6, menunjukkan efek batasan *ramp-rate* yang membuat nilai pembangkitan setiap generator semakin selektif. Hal ini berakibat nilia total biaya yang dikeluarkan akan lebih mahal $3386 dari pada total biaya dengan tidak melihat batasan *ramp-rate*. Hal ini dapat dilihat pada Tabel 4.31.

# BAB 5
# PENUTUP

## 5.1 Kesimpulan

Dari semua proses yang meliputi studi litealatur, serta simulasi dan analisis, maka terdapat beberapa hal yang dapat disimpullkan terkait Tugas Akhir ini, yaitu:

1. Aplikasi perhitungan DED dapat melakukan perhitungan sesuai dengan batasan tanpa melanggar *ramp-rate* yang telah ditentukan dengan menggunakan metode iterasi lambda dalam waktu tertentu, baik dengan memperhitungkan rugi-rugi yang didapatkan dalam formulasi *Bloss matrix* atau tanpa memperhitungkan rugi-rugi.
2. Adanya pengaruh batasan *ramp-rate* menyebabakan hasil perhitungan dalam DED semakin selektif. Dikarenakan nilai pembangkitan tidak bisa naik dan turun melebihi batasan *ramp-rate* yang telah ditetapkan pada setiap unit pembangkitan.
3. Dalam simulasi pengoperasisn aplikasi perhitungan DED, dengan adanya selektifitas penjadwalan optimum pembangkitan akibat batasan *ramp-rate* total biaya pembangkitan akan menjadi lebih mahal jika dibandingkan ketika melakukan perhitungan DED tanpa memperhatikan nilai *ramp-rate*. Hal ini terlihat dalam simulasi kasus 5 Tabel 4.23 memperlihatkan pengaruh batasan ramp-rate membuat total biaya operasi lebih mahal $2294, dan pada kasus 6 Tabel 4.31 dengan memperhatikan batasa *ramp-rate* lebih mahal $3386.
4. Dalam simulasi kasus 6 diperlukan penggantian nilai penjadwalan pembangkit, dikarenakan oleh pengaruh *ramp-rate* yang menyebab ketidakmampuan nilai optimal pembangkitan untuk melakukan pembangkitan optimal untuk periode beban berikutnya. Penggantian nilai pembangkitan dapat dilakukan dengan perhitungan *Economic Dispatch* pada satu interval waktu.
5. Hasil akhir dari aplikasi perhitungan yang dikembangkan dapat digunakan untuk meng-*upgrade* aplikasi perhitungan yang selama ini digunakan pada mata kuliah Operasi Optimum Sistem Tenaga Litrik

## 5.2 Saran

Adapun saran untuk penelitian selanjutnya yang berkaitan dengan Tugas Akhir ini, yaitu:

1. Aplikasi perhitungan DED dapat dikembangkan dengan memberikan perhitungan rugi-rugi dengan menggunakan *Optimal Power Flow (OPF),* agar hasil perhitungan lebih optimal.
2. Aplikasi DED dapat dikembanggkan dengan memberikan metode perhitungan yang berbeda sehingga dapat dibandingkan metode iterasi lambda.

# LAMPIRAN

**Tabel L.1 *Bloss matrix* Kasus 3**

| | Bij | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| | 1 | 0.49 | 0.14 | 0.15 | 0.15 | 0.2 |
| | 2 | 0.14 | 0.45 | 0.16 | 0.2 | 0.18 |
| 10*e-4 | 3 | 0.15 | 0.16 | 0.39 | 0.1 | 0.12 |
| | 4 | 0.15 | 0.2 | 0.1 | 0.4 | 0.14 |
| | 5 | 0.2 | 0.18 | 0.12 | 0.14 | 0.35 |

**Tabel L.1 *Bloss matrix* Kasus 4**

| Boo |
|---|
| 0.00154 |

| | Boi | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 10*e-5 | | 0.38 | 1.79 | -5.32 | 1.52 | 2.33 | 1.26 |

| | Bij | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 10*e-4 | 1 | 2.231 | 1.162 | -0.122 | -0.017 | 0.113 | 0.39 |
| | 2 | 1.162 | 1.89 | -0.077 | -0.048 | 0.069 | 0.28 |
| | 3 | -0.122 | -0.077 | 2.004 | -0.74 | -0.724 | -0.599 |
| | 4 | -0.017 | -0.048 | -0.74 | -1.479 | 0.538 | 0.342 |
| | 5 | 0.113 | 0.069 | -0.724 | 0.538 | 1.185 | 0.0053 |
| | 6 | 0.39 | 0.28 | -0.599 | 0.342 | 0.053 | 2.34 |

# Script Delphi

```
unit Unit31;

interface
uses  sysutils;

const

         max_units = 35;
         max_order = 10;
         max_curve_points = 10;
         max_period = 24;
         max_total_segments = 200;
         total_gen_tolerance = 0.01 ;
         ihr_tolerance = 0.000001 ;



         alpha : real = 1; {See note in loss matrix procedure}

type

         unit_array_real = array[1..max_units] of real;
         unit_name_array = array[1..max_units] of string;
         coefficients = array[0..max_order] of real;
         unit_poly_array = array[1..max_units] of coefficients;
         curve_points = array[0..max_curve_points] of real;
         unit_curve_array = array[1..max_units] of curve_points;
         system_ihr_array_real = array[1..max_total_segments] of real;
         system_ihr_array_integer = array[1..max_total_segments] of integer;
         B_matrix = array[1..max_units] of unit_array_real;
         filename_array = string;
         curvetype_list = (poly,pinc,pio);
         losstype_list = (noloss,constpf,lossform);
         solution_type_list = (lamsearch,tbllookup);
         schedtype_list = (totgen,totload);
         unit_array_period= array[1..max_period] of unit_array_real;
         period_array_real= array[1..max_period] of real;


var
        ioerr : boolean;
         ioval : integer;
     data_period:unit_array_period;      {Generation each unit on each period}
     total_period:period_array_real;     {total cost each period}
     genname:unit_name_array;            {Generator name identifier}
     p:unit_array_real;                  {Present value of P}
     pmin:unit_array_real;               {Minimum MW}
     pmax:unit_array_real;               {Maximum MW}
     UR:unit_array_real;
     DR:unit_array_real;
     maxup:unit_array_real;
     maxdown:unit_array_real;
     unitsebelum:unit_array_real;
     ramprate:boolean;
     minihr:unit_array_real;             {Minimum unit incremental heat rate}
     maxihr:unit_array_real;             {Maximum unit incremental heat rate}
     fuelcost:unit_array_real;           {Fuel cost ( $/fuel unit )}
     coeff : unit_poly_array;            {Unit polynomial coefficients}
     ihr_mwpoint:unit_curve_array;         {MW points on ihr cost curve}
     ihr_cost:unit_curve_array;            {Cost points for ihr curve}
     io_mwpoint : unit_curve_array;      {MW Points on unit io curve}
     io_cost : unit_curve_array;         {Cost points on io curve}
     minput:unit_array_real;             {Minimum input for PINC curves }
     penfac:unit_array_real;             {Loss penalty factor}
     penfacD:unit_array_real;             {Loss penalty factor}
     unitmin:unit_array_real;             {Initial Minimum Unit Generation}
     unitmax:unit_array_real;             {Initial Maximum Unit Generaion}
     totalcost:real;
     loadjam:unit_array_real;

     b00:real;                           {Loss matrix constant}
     b0:unit_array_real;                 {Loss matrix linear terms}
     b:B_matrix;                         {Loss matrix quadratic terms}
     seginccost:system_ihr_array_real;  {Segment inc cost for table look up }
     segunit:system_ihr_array_integer;  {Unit associated with segment}
     segmw:system_ihr_array_real;       {MW contributed by segment}
     order:system_ihr_array_integer;    {Order routine output}
     ordvalue:system_ihr_array_real;    {Numbers to be ordered}
     inputfile:text;
     filename:filename_array;
     title1, title2 : string[80];
     print_output, diagflag, read_data : boolean;
     inputchar,quitflag : char;
     linenumber, ngen : integer;
     mwlosses, lambda : real;
     curvetype_input, losstype_input : string[8];
     number_string : string[20];
     curveorder : integer;
```

```
     curvetype : curvetype_list;
     losstype : losstype_list;
     solution_type : solution_type_list;
     schedtype : schedtype_list;
     pgenmax, pgenmin, SRGenTotal : real;
     FF:Text;
     NoGenerator:integer ;
     NomerOrder:integer;
     ModeDataPembangkit:integer;
     ProsesRun:boolean;
     SR : real;
     numper : integer;
     jamstart : real;
procedure datainput(namafile :string);
procedure ihr_ftn(i : integer; unitmw : real;  var unitihr : real );
procedure GetFileName(s :string;var d:string;var f:string;var e :string);
procedure datadump( loadjam:real;unitsebelum:unit_array_real; var outfile:text );
procedure lambda_search_dispatch( schedmw:real; var lambda : real);
procedure loss_matrix_ftn;
procedure inverse_ihr_ftn(i : integer; unitihr : real; var unitmw : real );

procedure order_routine(        numorder : integer;
                                ordertable : system_ihr_array_real;
                        var     orderindex : system_ihr_array_integer );
procedure output_routine( schedmw:real; var outfile : text; lambda : real );
procedure prod_cost(        i : integer;
                        unitmw : real;
              var   unitcost : real );
procedure dataOutput(namafile :string);
PROCEDURE output_final( VAR  OUTFILE : TEXT );

function cekIoPoint(var unitG,Order:integer):boolean;
function cekIhrPoint(var unitG,Order:integer):boolean;
function CekEdcFile(filename:string): boolean;


implementation
function CekEdcFile(filename:string): boolean;
label keluar;
var s,d,f,e:string;
ff:text;
bc : boolean;
begin
bc:=false;
getfilename(filename,d,f,e);
 s:=trim(uppercase(f))+'.'+uppercase(e);
if s = 'EDC1.DAT' then bc:= true;
if s = 'EDCTEST.DAT' then bc:= true;

if s = 'EDC2.DAT' then bc:= true;
if s = 'EDC3.DAT'  then bc:= true;
if s = 'EDC4.DAT'  then bc:= true;
if s = 'EDC5.DAT'  then bc:= true;

if s = 'EXDED.DAT'  then bc:= true;
if s = 'EX3B.DAT'  then bc:= true;
if s = 'EX3C.DAT'  then bc:= true;
if s = 'EX3D.DAT'  then bc:= true;

if s = 'PR32.DAT'  then bc:= true;
if s = 'PR33.DAT'  then bc:= true;
if s = 'PR38.DAT'   then bc:= true;
if s = 'PR43A.DAT'  then bc:= true;
if s = 'PR43B.DAT'   then bc:= true;
if s = 'EX4D.DAT'   then bc:= true;

if bc = true then goto Keluar;

assign(ff,filename);
reset(ff);
readln(ff,s);
if pos('EDC FILE >>',s)>0 then bc:=true;
close(ff);


keluar:
cekedcfile:=bc;
end;
function cekIoPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIopoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (io_mwpoint[i,j+1] - io_mwpoint[i,j])<=0 then
      begin
        cekIoPoint:=false;
        UnitG:=i;
```

```
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;


function cekIhrPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIhrpoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (ihr_mwpoint[i,j+1] - ihr_mwpoint[i,j])<=0 then
      begin
        cekIhrPoint:=false;
        UnitG:=i;
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;

procedure prod_cost(       i : integer;
                     unitmw : real;
              var   unitcost : real );

      { Routine to return unit production cost given unit output in mw}
      { input : unit index = i}
      {         unit MW =  unitmw}
      { output: unit production cost = unitcost}

var
       j : integer;
       partmw, unitihr, segmentcost : real;

label return;

begin

case curvetype of
  poly :                          {Polynomial I/O curve}
      begin
      unitcost := 0;
      for j := curveorder downto 1 do
      unitcost := ( unitcost + coeff[ i,j ] ) * unitmw;

      unitcost := unitcost + coeff[ i,0 ];
      unitcost := unitcost * fuelcost[ i ];
      goto return
      end;

  pinc :                          {Piecewize incremental curve}
      begin
      unitcost := minput[ i ] * fuelcost[ i ];
      for j := 1 to curveorder do
         begin
         if (unitmw > ihr_mwpoint[ i,j ]) and ( j < curveorder ) then
                         {Calculate area under complete segment}
            begin
            segmentcost :=  ( (ihr_cost[i,j] + ihr_cost[i,j-1] )/2.0 ) *
                            ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) *
                                        fuelcost[ i ];
            unitcost := unitcost + segmentcost;
            end
            else
                         {Calculate area under partial segment}
            begin
            partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                              (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
            unitihr := ihr_cost[i,j-1] +
                           ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw;
            segmentcost :=  ( (unitihr + ihr_cost[i,j-1])/ 2.0 ) *
                             ( unitmw - ihr_mwpoint[i,j-1] )  *
                             fuelcost[ i ];
            unitcost := unitcost + segmentcost;

            goto return;
            end;
         end;

      end;

   pio :                          {Piecewize I/O curve}
```

```
            begin
            for j := 1 to curveorder do
               begin
               if io_mwpoint[i,j] > unitmw then
                  begin
                  partmw := (unitmw-io_mwpoint[i,j-1] ) /
                                 (io_mwpoint[i,j]-io_mwpoint[i,j-1] );
                  unitcost := io_cost[i,j-1] +
                                 ( io_cost[i,j]-io_cost[i,j-1] ) * partmw;
                  unitcost := unitcost * fuelcost[ i ];
                  goto return
                  end;
               if j = curveorder then     {Unit is at or above pmax}
                  begin
                  unitcost := io_cost[ i, j ] * fuelcost[ i ];
                  goto return
                  end;
               end;
            end;
      end;  { End of case statement}

   return:

   end; { End procedure }

   procedure output_routine( schedmw:real; var outfile : text; lambda : real );

   var
            limittxt : string[5];
            totalgen, totalload, totalmaxdown, totalmaxup : real;
            unitihr, unitinccost, unitcost : real;
            i : integer;

   label return;

   begin

   writeln(outfile);
   writeln(outfile,
         'generator   output    limit    maxup    maxdown     inc cost penalty fact   operating
cost');
   writeln(outfile,
            '                    mw                mw        mw         $/mwhr                      $/hr
');
   writeln(outfile,
         '---------  ------   -----  -------- --------  ---------- -----------   ------------
-');

   totalgen := 0.0;
   totalcost := 0.0;
   totalmaxup:=0;
   totalmaxdown:=0;


   for i := 1 to ngen do
      begin
      write(outfile,genname[i]:9);
      write(outfile, ' ',p[i]:6:1, '    ');
      limittxt :=  '     ';
      if abs( p[i] - pmin[i] ) < total_gen_tolerance then limittxt := 'min  ';
      if abs( p[i] - pmax[i] ) < total_gen_tolerance then limittxt := 'max  ';
      write(outfile, limittxt );

      ihr_ftn( i, p[ i ], unitihr) ;    {Get unit incremental heat rate}

      unitinccost := unitihr * fuelcost[i];
      write(outfile, maxup[i]:10:3,'');
      write(outfile, maxdown[i]:10:3,'');
      write(outfile, unitinccost:9:4);
      write(outfile, '    ',penfac[i]:9:4, '    ');

      prod_cost( i, p[ i ], unitcost );          {Calculate unit operating cost}

      writeln(outfile, '   ',unitcost:9:2);
      totalgen := totalgen + p[i];
      totalcost := totalcost + unitcost;
      totalmaxup:= totalmaxup + maxup[i];
      totalmaxdown:= totalmaxdown + maxdown[i];
      end;
   writeln(outfile,
            '---------  ------                                                                  ---------
-----');
   write(outfile, ' totals');
   write(outfile, totalgen:9:1,
   '                                                        ', totalcost:9:2);
   writeln(outfile);
   writeln(outfile, ' lambda = ', lambda:10:4 );
   writeln(outfile);

   if (schedtype = totgen ) and ( losstype <> lossform ) then goto return;
```

```
if schedtype=totload then totalload := schedmw;

if schedtype=totgen then  totalload := totalgen - mwlosses;

writeln(outfile, 'total load = ',totalload:10:1,
                 ' total losses = ',mwlosses:10:1);
writeln(outfile);
writeln(outfile, 'maxup for next periode = ',totalmaxup:10:3);
writeln(outfile);
writeln(outfile, 'maxdown for next periode = ',totalmaxdown:10:3);


return:

end; { End procedure }

PROCEDURE output_final( VAR  OUTFILE : TEXT );

VAR I,J : INTEGER;

BEGIN


      WRITELN(OUTFILE);

      WRITELN(OUTFILE,'FINAL OUTPUT DYNAMIC ECONOMIC DISPATCH ');
      WRITELN (OUTFILE);
      WRITELN(OUTFILE,'NOTE:(PERIOD 1 NOT ALWAYS CALCULATE FOR LOAD AT HOUR 1)');
      WRITELN (OUTFILE);
      WRITE(OUTFILE,'PERIOD       UNIT STATUS ');
      FOR I:=1 TO ngen*8-11 DO WRITE(OUTFILE,' ');
      WRITELN(OUTFILE,' PCOST     LOAD');
      WRITE(OUTFILE,'       ');
      FOR J := 1 TO ngen DO WRITE(OUTFILE,J:8);
      WRITELN(OUTFILE,'         R/HR      MW');
      FOR J := 1 TO 34+ngen*8 DO WRITE(OUTFILE,'-');
      WRITELN(OUTFILE);
      FOR i := 1 TO numper DO
      BEGIN
        WRITE(OUTFILE,i:4,'    ');
        FOR J := 1 TO ngen DO
          WRITE(OUTFILE,' ',data_period[i,j]:6:1,' ');
        WRITELN(OUTFILE,' ',total_period[i]:9:2,' ',loadjam[i]:8:1);
        {OPTSTATE := PATH[ OPTSTATE ];   }
      END;

END;



procedure order_routine(        numorder : integer;
                               ordertable : system_ihr_array_real;
                         var   orderindex : system_ihr_array_integer );

{ subroutine to order a list, least first                                  }
{                                                                          }
{ input numorder = the number of items to be ordered                       }
{ input ordertable = the items to be ordered                               }
{ output orderindex = pointer to order value table                         }
{                                                                          }
{ nxt = Table used in order subroutine                                     }
{                                                                          }
  var
      stop:boolean;
      i,j,top,last,indx:integer;
      nxt : system_ihr_array_integer;

  begin
    for i := 1 to numorder do begin
      if (i <= 1) then begin
        top := 1;
        nxt[ 1 ] := 0;
        end
      else begin
        j := top;
        last := 0;
        repeat
          stop := true;
          if (ordertable[ i ] > ordertable[ j ]) then begin
            last := j;
            j := nxt[ j ];
            stop  := (j = 0);
            if (stop) then begin
              nxt[ last ] := i;
              nxt[ i ] := 0;
            end
            end
          else begin
            if (j <> top) then begin
```

```
              nxt[ last ] := i;             { j not = top }
              nxt[ i ] := j;
              end
            else begin
              top := i;                     { j = top }
              nxt[ i ] := j;
            end;
          end;
        until stop;
      end;
    end;
    indx := 1;
    j := top;
    repeat
      orderindex[ indx ] := j;
      j := nxt[ j ];
      indx := indx + 1;
    until (j = 0);
  end;


procedure inverse_ihr_ftn(       i : integer;
                           unitihr : real;
                       var unitmw : real );

      { Routine to return unit MW given unit incremental heat rate}
      { input : unit number = i}
      {         unit inc heat rate = unitihr}
      { output: unit MW stored in unitmw}

label  return;

var
         unitihr1, delihr, partihr, dihrdp : real;
         segmentihr : real;
         j, step : integer;

begin

if unitihr >= maxihr[ i ] then
    begin
    unitmw := pmax[ i ];
    goto return
    end;
if unitihr <= minihr[ i ] then
    begin
    unitmw := pmin[ i ];
    goto return
    end;

case curvetype of
  poly :                        {Polynomial curve}
      begin
      if curveorder <= 1 then
       begin
       if unitihr > coeff[ i,1 ] then unitmw:=pmax[i] else unitmw:=pmin[i];
       goto return
       end;

      if curveorder = 2 then
       begin
       unitmw := ( unitihr - coeff[ i,1 ] ) / ( 2.0 * coeff[ i,2 ] );
       goto return
       end;

            { for curves of order >= 3 search for unitmw using Newtons method }

      unitmw := ( pmin[ i ] + pmax[ i ] )/ 2.0;
      step := 0;
      repeat
      step := step + 1;

      unitihr1 := 0;                    {Calc unitihr at unitmw as unitihr1}
      for j := curveorder downto 2 do
         unitihr1 := ( unitihr1 + j * coeff[i,j] ) * unitmw;
      unitihr1 := unitihr1 + coeff[i,1];
      delihr := unitihr - unitihr1;
      if abs( delihr ) < ihr_tolerance then goto return;

      dihrdp := 0;                       {Calc curve second derivative}
      for j := curveorder downto 3 do
         dihrdp := ( dihrdp + j*(j-1) * coeff[i,j] ) * unitmw;
      dihrdp := dihrdp + 2.0 * coeff[ i,2 ];
      unitmw := unitmw + delihr/dihrdp;

      until step > 35;

      goto return
      end;
```

```
  pinc :                              {Piecewize incremental curve}
      begin
      j := 0 ;
      repeat
      j := j + 1;
      until (ihr_cost[i,j] > unitihr) or
            (j = curveorder);

      partihr := ( unitihr - ihr_cost[i,j-1] )/
                           ( ihr_cost[i,j] - ihr_cost[i,j-1] );
      unitmw := ihr_mwpoint[i,j-1] +
                  ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) * partihr;
      goto return
      end;

    pio :                             {Piecewise I/O curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      if j = curveorder then
         begin
         unitmw := io_mwpoint[i,j];
         goto return
         end;
      segmentihr :=(io_cost[i,j] - io_cost[i,j-1]) /
                              (io_mwpoint[i,j] - io_mwpoint[i,j-1]);
      until segmentihr >= unitihr;

      unitmw := io_mwpoint[ i,j-1 ];
      goto return
      end;

  end;  { End of case statement}

return:

end; { End procedure }

procedure loss_matrix_ftn;

      { Routine to calculate losses and penalty factors from loss formula}
      {  Input:   Table of unit generation p[ i ]}
      {           Loss formula b( i,j ), b0[ i ], b00}
      {  Output:  losses mwlosses and penalty factors penfac[ i ]}
      {  Note: loss formula expects p(i)'s to be in per unit so divide by 100.}

var
         i, j : integer;
         incloss, penfac_old, penfac_new : real;

label return;

begin

      mwlosses := b00;
      for i := 1 to ngen do
         begin
         mwlosses := mwlosses + b0[i] *
                        ( p[i]/100.0) + b[i,i] * sqr( p[i]/100.0 );
         for j := i+1 to ngen do
           mwlosses := mwlosses + 2.0 * ( p[i]/100.0) * ( p[j]/100.0 ) * b[i,j]
         end;

      mwlosses := mwlosses * 100.0;


      for i := 1 to ngen do
         begin
         penfac_old := penfac[ i ];
         incloss := b0[i];
         for j := 1 to ngen do
            incloss := incloss + 2.0 * ( p[j]/100.0 ) * b[i,j];
         penfac_new := 1.0 / ( 1.0 - incloss );
         penfac[ i ] := penfac_old + alpha *
                                        ( penfac_new  -  penfac_old)
         end;

{ Note, in the formula above the penalty factor is "filtered" by the }
{ alpha filtering constant. If alpha is set to 1.0 no filtering action }
{ takes place, if alpha is 0.0 penfac is constant at 1.0 , suggested }
{ value for alpha is 0.5 to 0.9 }

return:

end; { End procedure }


procedure lambda_search_dispatch( schedmw:real; var lambda : real);
```

```
        var
                i, n, lossiter : integer;
                lambdamin, lambdamax : real;
                lambdastart, deltalambda, targetgen : real;
                unitihr, unitmw, totalgen, totalmaxup, totalmaxdown :  real;
                endloop : boolean;

        begin

        for i := 1 to ngen do                    { Set unit output to midrange}
          begin
          p[ i ] := ( pmin[ i ] + pmax[ i ] ) / 2.0
          end;

        lossiter := 0;
        endloop := false;

        repeat                                    {Top of iterative loop with losses}

          lambdamin := 10000.0;
          lambdamax := 0.0;
          mwlosses := 0 ;
          if losstype = lossform then   { Calc losses and pen factors}
          begin
            loss_matrix_ftn;
            writeln(ff);
            writeln(ff,' mw losses = ',mwlosses:10:1);
          end;

          for i := 1 to ngen do                  {Calculate max and min lambdas}
            begin
            ihr_ftn (i,pmax[i],maxihr[i]);
            lambda := maxihr[ i ] * penfac[ i ] * fuelcost[ i ];
            if lambda > lambdamax then lambdamax := lambda;
            ihr_ftn (i,pmin[i],minihr[i]);
            lambda := minihr[ i ] * penfac[ i ] * fuelcost[ i ];
            if lambda < lambdamin then lambdamin := lambda
            end;


            writeln(ff,' lambda limits = ',lambdamin:10:4,lambdamax:10:4);


          lambdastart := ( lambdamax + lambdamin ) / 2.0;
          deltalambda := ( lambdamax - lambdamin ) / 2.0;


           writeln(ff,' lambdastart deltalambda = ',lambdastart:10:4,deltalambda:10:4);
          {Set up total generation target}
          if schedtype = totgen then targetgen := schedmw;
          if schedtype = totload then targetgen := schedmw + mwlosses;
          {Lambda search}
          lambda := lambdastart;
          writeln(ff, ' targetgen = ',targetgen:10:1);

          n := 0;
          repeat                              {Top of lambda search loop}
              n := n + 1;
              totalgen := 0;
              totalmaxup:=0;
              totalmaxdown:=0;

              for i := 1 to ngen do
                 begin
                 unitihr := lambda / ( penfac[ i ] * fuelcost[ i ] )  ;
                 inverse_ihr_ftn( i, unitihr, unitmw );  {For given unitihr get unitmw}
                 p[ i ] := unitmw;
                 maxup[i]:=p[i]+UR[i];
                   if maxup[i]>unitmax[i] then maxup[i]:=(unitmax[i]-p[i]) else maxup[i]:=UR[i] ;
                 maxdown[i]:=p[i]-DR[i];
                   if     maxdown[i]<unitmin[i]     then     maxdown[i]:=(p[i]-unitmin[i])     else
maxdown[i]:=DR[i];
                 totalgen := totalgen + p[ i ];
                 totalmaxup:=totalmaxup + UR[i];
                 totalmaxdown:=totalmaxdown + DR[i];

                 end;


                 writeln(ff,' lambda = ',lambda:2:4,'  totalgen = ',totalgen:10:3,' delta lambda
= ',deltalambda:1:4);


              if abs( totalgen - targetgen ) >= total_gen_tolerance then
                begin
                if totalgen > targetgen then lambda := lambda - deltalambda;
                if totalgen < targetgen then lambda := lambda + deltalambda;
                deltalambda := deltalambda / 2.0
```

```
        end;

      until ( abs( totalgen - targetgen ) < total_gen_tolerance ) or
            ( n > 35 ) ;

{See if another loss iteration is needed}

      if losstype <> lossform then endloop := true;
      lossiter := lossiter + 1;
      if lossiter > 10 then endloop := true ;
      if  abs( totalgen - targetgen ) > total_gen_tolerance  then  ProsesRun:=false;
      //else ProsesRun:=True;        //check if the program really get the right result

until endloop;


end; { End Lambda search procedure }

procedure GetFileName(s :string;var d:string;var f:string;var e :string);
var ss ,sd:string;
  n:integer;
begin
  ss := s;
  d := '';
  n :=  pos('\',ss);
  while  n>0 do
  begin
    sd := copy(ss,1,n);
    d := d+sd;
    delete(ss,1,n);
  n :=  pos('\',ss);
  end;
  n := pos('.',ss);
  if n = 0 then
  begin
     f := ss;
     e:= '';
  end else
  begin
     f:=copy(ss,1,n-1);
     delete(ss,1,n);
     e := ss;
  end;
end;
procedure IOCheck( linenumber : integer );
begin
  IOVal := IOresult;
  IOErr := (IOVal <> 0);
end; { of proc IOCheck }


procedure datainput(namafile :string);

label quit;

var i,j,k,jj : integer;
    a : char;

begin
print_output := True;

linenumber := 0;

assign(inputfile, namafile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                    }
{ Open data file     }
{                    }
reset(inputfile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                    }
{ Read file header }
{                    }
linenumber := linenumber + 1;
readln( inputfile, title1 );
     iocheck( linenumber );
     if ioerr then goto quit;
linenumber := linenumber + 1;
readln( inputfile, title2 );
     iocheck( linenumber );
     if ioerr then goto quit;

     {                                                      }
     {  Read Number of generators, curve type, loss type  }
     {                                                      }
linenumber := linenumber + 1;
read( inputfile, ngen );
     iocheck( linenumber );
```

```
     if ioerr then goto quit;

repeat
read( inputfile, inputchar );
        iocheck( linenumber );
        if ioerr then goto quit;
until inputchar <> ' ';
curvetype_input := inputchar;
repeat
read( inputfile, inputchar );
      iocheck( linenumber );
      if ioerr then goto quit;
if inputchar <>' ' then curvetype_input := curvetype_input + inputchar;
until inputchar = ' ';

read( inputfile, curveorder );
      iocheck( linenumber );
      if ioerr then goto quit;

repeat
read(inputfile, inputchar );
     iocheck( linenumber );
     if ioerr then goto quit;
until inputchar <> ' ';
losstype_input := inputchar;
readln(inputfile);

 {                                                      }
 {  Set up internal variables for curvetype and losstype  }
 {                                                      }

if (curvetype_input = 'poly') or
   (curvetype_input = 'POLY') then curvetype := poly;

if (curvetype_input = 'pinc') or
   (curvetype_input = 'PINC') then curvetype := pinc;

if (curvetype_input = 'pio' ) or
   (curvetype_input = 'PIO' ) then curvetype := pio;

if (losstype_input = 'n'  ) or
   (losstype_input = 'N'  ) then losstype := noloss;

if (losstype_input = 'c'  ) or
   (losstype_input = 'C'  ) then losstype := constpf;

if (losstype_input = 'l'  ) or
   (losstype_input = 'L'  ) then losstype := lossform;


         {                      }
         { Read generator data }
         {                      }

for i := 1 to ngen do
   begin  { Read generator name }
      linenumber := linenumber + 1;
      repeat
      read( inputfile, inputchar );
            iocheck( linenumber );
            if ioerr then goto quit;
      until inputchar <>  ' ';
      genname[i] := inputchar;
      repeat
      read( inputfile, inputchar );
            iocheck( linenumber );
            if ioerr then goto quit;
      if inputchar <>' ' then genname[i] := genname[i] + inputchar;
      until inputchar = ' ';
         {                                    }
         { Read generator min, max, fuelcost }
         {                                    }
      readln( inputfile, pmin[i], pmax[i], fuelcost[i], DR[i], UR[i] );
            iocheck( linenumber );
            if ioerr then goto quit;
         {                                    }
         { Read generator cost curve data    }
         {                                    }
      case curvetype of
        poly :
               begin  { read polynomial curve data}
               for j := 0 to curveorder do
                 begin
                   linenumber := linenumber + 1;
                   readln( inputfile, coeff[i,j] );
                    iocheck( linenumber );
                    if ioerr then goto quit;
                 end;
               end;
```

```
        pinc :
               begin  { read piecewise incremental cost curve data}
               linenumber := linenumber + 1;
               readln( inputfile, minput[i] );
                 iocheck( linenumber );
                 if ioerr then goto quit;
               for j := 0 to curveorder  do
                 begin
                   linenumber := linenumber + 1;
                   readln( inputfile, ihr_mwpoint[i,j], ihr_cost[i,j] );
                     iocheck( linenumber );
                     if ioerr then goto quit;
                 end;
               end;

        pio  :
               begin  { read piecewise I/O curve data}
               for j := 0 to curveorder do
                 begin
                   linenumber := linenumber + 1;
                   readln( inputfile, io_mwpoint[i,j], io_cost[i,j] );
                     iocheck( linenumber );
                     if ioerr then goto quit;
                 end;
               end;

        end;  { End of case statement}
    end;
           {                  }
           { Read loss data }
           {                  }

case losstype of

  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0
        end;

  constpf  :
      begin   { read constant penalty factor data}
        linenumber := linenumber + 1;
        for i := 1 to  ngen do
         begin
         if i < ngen then
         read( inputfile, penfac[i] )
         else
         readln( inputfile, penfac[ngen] );
            iocheck( linenumber );
            if ioerr then goto quit
         end;
      end;

  lossform :

      begin  { read loss formula data}

      linenumber := linenumber + 1;
      readln( inputfile, b00 );
            iocheck( linenumber );
            if ioerr then goto quit;

      linenumber := linenumber + 1;
      for j := 1 to  ngen  do
        begin
        if j < ngen then
            read( inputfile, b0[ j ] )
           else
           readln( inputfile, b0[ngen] );
            iocheck( linenumber );
            if ioerr then goto quit;
        end;

      for i := 1 to ngen do
        begin
        linenumber := linenumber + 1;
        for j := 1 to ngen  do
         begin
         if j < ngen then
            read( inputfile, b[ i, j ] )
            else
            readln( inputfile, b[ i, ngen ] );
            iocheck( linenumber );
            if ioerr then goto quit
         end;
        end;
        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0
      end;
```

```
  end;  { End of case statement}
{                                 }
{  End of data input, close file  }
{                                 }

close ( inputfile );

{A very useful table is the incremental cost at the max and min of each unit. }
{These are calculated and stored in tables maxihr and minihr.}
{Also calculate the max and min generation}



quit:

end;   { end of data input }




procedure dataOutput(namafile :string);
var i,j,k,jj : integer;
    a : char;
    d,e,f:string;
   var inputfile:text;
begin
getfilename(namafile,d,f,e);
assign(inputfile, namafile);
rewrite(inputfile);
writeLn( inputfile, 'EDC FILE >> '+f+'.'+e );
writeln( inputfile, '====================================');
write( inputfile, ngen );
write(inputfile,' ');

if  curvetype = poly then write(inputfile,'POLY');
if  curvetype = pinc then write(inputfile,'PINC');
if  curvetype = PIO then write(inputfile,'PIO');
write(inputfile,' ');
write(inputfile,curveorder);
write(inputfile,' ');

if losstype = noloss  then write(inputfile,'NOLOSS');
if losstype = constpf  then write(inputfile,'CONSTPF');
if losstype = lossform  then write(inputfile,'LOSSFORM');
writeln(inputfile);
for i := 1 to ngen do
   begin
     write(inputfile,genname[i]);
     write(inputfile,' ');
     writeln(  inputfile,  pmin[i]:15:6,  pmax[i]:15:6,  fuelcost[i]:15:6,  DR[i]:15:6,
UR[i]:15:6 );
      case curvetype of
          poly :
                begin
                for j := 0 to curveorder do
                  begin
                    writeln( inputfile, coeff[i,j]:15:6 );
                  end;
                end;

          pinc :
                begin
                writeln( inputfile, minput[i]:15:6 );
                for j := 0 to curveorder  do
                  begin
                    writeln( inputfile, ihr_mwpoint[i,j]:15:6, ihr_cost[i,j]:15:6 );
                  end;
                end;

          pio  :
                begin
                for j := 0 to curveorder do
                  begin
                    writeln( inputfile, io_mwpoint[i,j]:15:6, io_cost[i,j]:15:6);
                  end;
                end;

          end;
   end;

case losstype of

  noloss :
        begin
        for i := 1 to ngen do      { Init penalty factors}
            penfac[ i ] := 1.0
        end;

  constpf  :
```

```
      begin
         for i := 1 to  ngen do
         begin
         if i < ngen then
         begin
         write( inputfile, penfac[i]);
         write(inputfile,' ');

         end
         else writeln( inputfile, penfac[ngen]);
         end;
      end;

  lossform :

      begin

      linenumber := linenumber + 1;
      writeln( inputfile, b00);
       for j := 1 to  ngen  do
        begin
        if j < ngen then
           begin
           write( inputfile, b0[ j ]);
           write(inputfile,' ');
           end
           else
           writeln( inputfile, b0[ngen]);
        end;

      for i := 1 to ngen do
        begin
        for j := 1 to ngen  do
         begin
         if j < ngen then
            begin
            write( inputfile, b[i, j]);
            write(inputfile,' ');

            end
            else
            writeln( inputfile, b[i,ngen]);
         end;
        end;

        for i := 1 to ngen do      { Init penalty factors}
              penfac[ i ] := 1.0

      end;

  end;


close ( inputfile );



end;

procedure ihr_ftn(          i : integer;
                    unitmw : real;
                var unitihr : real );

      { Routine to return unit incremental heat rate given unit output in MW }
      { input : unit index = i}
      {         unit mw = unitmw}
      { output: unit incremental heat rate = unitihr}

var
         partmw : real;
         j : integer;

begin

case curvetype of
  poly :                    {Polynomial I/O curve}
      begin
      unitihr := 0.0;
      for j := curveorder downto 2 do
        unitihr := ( unitihr + j * coeff[ i,j ] ) * unitmw;
      unitihr := unitihr + coeff[ i,1 ]
      end;

  pinc :                     {Piecewize incremental curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      until (ihr_mwpoint[ i,j ] > unitmw) or (j = curveorder);
```

```
            partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                      (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
            unitihr := ihr_cost[i,j-1] + ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw
            end;

          pio  :                      {Piecewize I/O curve}
            begin
            j := 0;
            repeat
            j := j + 1;
            until (io_mwpoint[ i,j ] > unitmw) or (j = curveorder);

            unitihr := (io_cost[i,j] - io_cost[i,j-1] ) /
                              ( io_mwpoint[i,j] - io_mwpoint[i,j-1] )
            end;
          end;  { End of case statement}


        end; { End ihr_ftn procedure }

        procedure datadump( loadjam:real; unitsebelum:unit_array_real ;var outfile:text );


        var
           i,j: integer;

        begin
        //if jamstart < 1 then jamstart:=1;
        for i := 1 to ngen do
        begin
        pmax[i]:=unitmax[i]-(SR/100)*unitmax[i];
        pmin[i]:=unitmin[i];
        if unitsebelum[i] <> 0 then if ramprate then
          begin
          if     (unitsebelum[i]+UR[i])<(unitmax[i]-(SR/100)*unitmax[i])     then     pmax[i]     :=
unitsebelum[i]+UR[i];
          if (unitsebelum[i]-DR[i])>(unitmin[i]) then pmin[i] := unitsebelum[i]-DR[i];
          end;
        end;

        writeln(outfile);
        writeln(outfile, title1);
        writeln(outfile, title2);
        writeln(outfile);
        writeln(outfile ,' number of generator units = ',ngen );

        case curvetype of
          poly : writeln(outfile ,' unit curve type = poly ');
          pinc : writeln(outfile ,' unit curve type = pinc ');
          pio  : writeln(outfile ,' unit curve type = pio');
          end;  { End of case statement}

        writeln(outfile ,' curve order = ',curveorder);

        case losstype  of
          noloss :   writeln(outfile ,' network loss representation = noloss ');
          constpf :  writeln(outfile ,' network loss representation = constpf ');
          lossform : writeln(outfile ,' network loss representation = lossform ');
          end;  { End of case statement}
        for i := 1 to ngen do
          begin
           writeln(outfile);
           write(outfile, genname[i],' limits = ',pmin[i]:7:2, ' ',pmax[i]:7:2 );
           writeln(outfile, ' fuelcost = ',fuelcost[i]:10:4 );
             case curvetype of
                  poly :
                      begin
                      writeln(outfile,'  polynomial coefficients' );
                      for j := 0 to curveorder do
                        begin
                        writeln(outfile, coeff[i,j]:15:6);
                        end;
                      end;
                  pinc :
                      begin
                      writeln(outfile,' incremental cost curve points');
                      writeln(outfile,'input at pmin = ',minput[i]:10:2);
                      for j := 0 to curveorder do
                        begin
                        writeln(outfile,ihr_mwpoint[i,j]:9:2,ihr_cost[i,j]:9:3 )
                        end;
                      writeln(outfile);
                      end;
                  pio :
                      begin
                      writeln(outfile,' cost curve points');
                      for j := 0 to curveorder do
                        begin
                        writeln(outfile,io_mwpoint[i,j]:9:2,'  ',io_cost[i,j]:9:3 )
                        end;
```

# DAFTAR GAMBAR

Institut
Teknologi
Sepuluh Nopember

**FINAL PROJECT – TE 141599**

# DYNAMIC ECONOMIC DISPATCH (DED) CONSIDERING RAMP-RATE USING LAMBDA ITERATION BASED ON DELPHI

**Dwi Haryanto**
**NRP 2211100158**

**Advisor**
**Prof. Ir. Ontoseno Penangsang, M. Sc, Ph. D.**
**Ir. Sjamsjul Anam, MT.**

**ELECTRICAL ENGINEERING DEPARTEMENT**
**Faculty of Industrial Technology**
**Institut Teknologi Sepuluh Nopember**
**Surabaya 2015**

# ABSTRAK

**_Dynamic Economic Dispatch_ (DED) dengan Memperhatikan _Ramp-rate_ Menggunakan Metode Iterasi Lambda Berbasis Delphi**

Hardi Rizkyanto
2211100143

Dosen pembimbing 1 :Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dosen Pembimbing 2 :Ir. Sjamsjul Anam, MT.

**Abstrak**

Pembagian pembebanan pembangkit yang bertujunan untuk meminimalkan biaya pembangkitan dikenal dengan istilah *Economic Dispatch* (ED). Perubahan beban yang dilihat dalam setiap interval waktu akan menghasilkan perhitungan *Dynamic Economic Dispatch* (DED). Pada Tugas Akhir ini memperhatikan *ramp-rate* pada perhitungan DED. *Ramp-rate* digunakan untuk merubah batasan yang akan digunakan dalam perhitungan DED pada interval waktu berikutnya, sehingga pembebanan yang diberikan kepada setiap pembangkit akan semakin selektif. Tugas Akhir menggunakan metoda iterasi lambda sebagai penyelesaian masalah optimalisasi biaya pada DED. Metoda iterasi lambda akan diterapkan pada aplikasi pemrograman Delphi untuk meng-*upgrade* aplikasi perhitungan PowerGen yang digunakan pada matakuliah Optimalisasi Sistem Tenaga Elekto ITS, dengan cara menambahkan fitur aplikasi perhitungan DED didalam *software*. Uji kebenaran akan didapatkan dengan melihat hasil pengoperasian aplikasi perhitungan yang telah dibuat tidak melanggar batasan *ramp-rate.*

**Kata kunci**: *Economic Dispatch, Dynamic Economic Dispatch, ramp-rate,* Delphi

# ABSTRACT

***Dynamic Economic Dispatch (DED) Considering Ramp-Rate Using Lambda Iteration Based on Delphi***

Hardi Rizkyanto
2211100143

Dosen pembimbing 1 :Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D
Dosen Pembimbing 2 :Ir. Sjamsjul Anam, MT.

**Abstract**

Load-sharing power plants which aims to minimize the cost of the generation known as the Economic Dispatch (ED). Load changes seen in each time interval will result in the calculation of Dynamic Economic Dispatch (DED). In this final notice on the ramp-rate calculation DED. Ramp-rate is used to change the limit to be used in the calculation of DED in the next time interval, so that the load given to each generation will be more selective. The final task using lambda iteration method as cost optimization problem solving at DED. Lambda iteration method will be applied to the Delphi programming application to upgrade the PowerGen computing applications used on subjects Optimization of Power System Elekto ITS, by adding features DED calculation within software applications. Testing will be obtained by looking at the operation of the application calculations that have been made do not violate more than ramp-rate limit

**Index Term:** Economic Dispatch, Dynamic Economic Dispatch, ramp-rate, Delphi

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT atas segala rahmat, karunia, dan petunjuk yang telah dilimpahkan-Nya sehingga penulis mampu menyelesaikan Tugas Akhir dengan judul :

***Dynamic Economic Dispatch* (DED) dengan Memperhatikan *Ramp-rate* Menggunakan Metode Iterasi Lambda Berbasis Delphi**

Tugas Akhir ini disusun sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan S1 pada Bidang Studi Teknik Sistem Tenaga, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.

Pada kesempatan ini penulis hendak menyampaikan rasa terima kasih kepada pihak-pihak yang memberikan peranan penting dalam menyelesaikan Tugas Akhir ini, kepada:

1. Allah SWT atas limpahan Rahmat dan Petunjuk-Nya serta Nabi Muhammad SAW atas tuntunan jalan-Nya.
2. Bapak dan Ibu yang telah membesarkan, mendidik saya hingga dewasa kini.
3. Prof. Ir. Ontoseno Penangsang, M.Sc, Ph.D dan Dr. Rony Seto Wibowo, ST., MT. sebagai dosen pembimbing yang telah memberikan arahan dan perhatiannya dalam Tugas Akhir ini.
4. Seluruh dosen yang telah memberikan ilmunya selama kuliah, karyawan, dan keluarga besar Jurusan teknik Elektro ITS
5. Teman-teman Teknik Elektro ITS 2011 dan khususnya kepada teman-teman satu kelompok atas bantuan kalian selama masa pengerjaan Tugas Akhir ini
6. Semua pihak yang telah membantu baik secara langsung maupun tidak langsung, yang tidak mungkin saya sebutkan satu per satu

Penulis menyadari bahwa Tugas Akhir ini belum sempurna, Oleh karena itu saran dan masukan sangat diharapkan untuk perbaikan dimasa yang akan datang.

Surabaya, Juli 2015

Penulis

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR PUSTAKA


[1] Benhima, F, "*Solving Dynamic Economic Load Dispatch Witg Ramp rate Limit Using Quadratic Programming*", IEEE 978-1-4799-1255-1/13, 2013.

[2] Rabiee, Abbas, "*Fast Dynamic Economic Power Dispatch Problems Solution Via Optimality Condition Decomposition*", IEEE Transaction on Power System, Vol. 29, No. 2, March 2014.

[3] Allen J. Wood, Bruce F. Wollenberg, "Power, Generation, Operation, and Control", John Willey & Sons Inc, America, 1996.

[4] Kusnassriyanto, "*Belajar Pemrograman Delphi",* MODULA, Bandung, 2011.

[5] Saadat, Hadi, "*Power System Analysis 2nd Edition",* McGrowHill, Ch.1, 1999.

[6] Caelho, Leandro and Lee,Chu-Sheng, "*Solving Economic Load Dispatch Problem in Power Systems Using Chaotic and Gaussian Particle Swarm Optimization Approaches*", ELSEVIER Electrical Power and Energy Systems, Vol. 30, pp. 297-307, 2008.

[7] El-Harawary, M. E, "Electrical Power System: Design and Analysis-Rev Printing", IEEE Press Power System Enginering Series, United States of America, 1995.

[8] Mahatmya, Atya, "Implementasi Algoritma Ant Colony Optimization untuk Menyelesaikan Persamalahan Dynamic Economic Dispatch dengan Memperhatikan Rugi-rugi Daya Transmis dan Valve Point Effect", Jurusan Teknik Elektro FTI-ITS, Surabaya, 2013.

[9] Benhima, F, "Constrained Dynamic Economical Dispatch Using a Compact Quadratic Programming Method Including Losses", The International Conference on Electronics and Oil: From Theory to Aplication, Algeria, March 2013.

[10] Kumar, Pardeeep, "*Dynamic Economic Dispatch Using Defferentila Evolution Algorithm",* Department of Electrical and Instrumentation Enginering Thapar University, Punjab, 2013.

*Halaman ini sengaja dikosongkan*

# RIWAYAT HIDUP PENULIS

**HARDI RIZKYANTO,** lahir di Bojonegoro, 26 Juli 1993. Penulis tamat dari bangku sekolah dasar di SDN Simomulyo 4 Surabaya tahun 2005 dan melanjutkan di sekolah menengah pertama di SMP Al-Hikmah Surabaya, lulus tahun 2008. Setelah lulus SMP, penulis melanjutkan sekolah ke SMAN 9 Surabaya. Pada tahun 2011, penulis melanjutkan studi S1 di Institut Teknologi Sepuluh Nopember (ITS) Surabaya jurusan Teknik elektro dan mengambil konsentrasi dalam Bidang Studi Teknik Sistem Tenaga. Putra pertama dari dua bersaudara dari orang tua Bapak M. Hari Poernama dan Ibu Boedi Widyaningsih ini aktif dalam berbagai kegiatan, diantaranya HIMATEKTRO ITS, EEVENT ELEKTRO ITS. Penulis dapat dihubungi melalui alamat email : hardirizky@gmail.com

# BAB 1
# PENDAHULUAN

## 1.1 Latar Belakang

Pengaturan penjadwalan pembangkit dibutuhkan oleh perusahaan untuk mendapatkan hasil perhitungan yang paling ekonomis sehingga perusahaan tidak mengalami kerugian. *Economic Dispatch* (ED) merupakan pembagian pembebanan pada unit-unit pembangkit yang ada dalam sistem secara optimal ekonomi pada harga beban sistem tertentu. Dengan penerapan *Economic Dispatch* maka akan didapatkan biaya pembangkitan yang minimum terhadap produksi daya listrik yang dibangkitkan unit-unit pembangkit pada suatu system kelistrikan. Pada Tugas Akhir ini memodifikasi persamaan yang ada pada ED berdasarkan perubahan beban di tiap waktunya. Persamaan tersebut dikenal sebagai *Dynamic Economic Dispatach* (DED) dengan memperhatikan *ramp-rate* yang sangat di pengaruhi oleh berbagai sistem yang terpasang di dalam pembangkit tersebut. Sebagai contoh, pada pembangkit dengan turbin uap berbahan bakar batu bara mempunyai level *ramp-rate* yang berbeda. Pengaruh dari *ramp-rate* yang merubah batasan yang ada pada persamaan untuk mendapatkan nilai optimal. Dalam referensi [1][2], menggunakan persamaan *ramp-rate* yang merupakan modifikasi pada batasan *maximum* dan *minimum* pada tiap unit berdasarakan perubahan beban, yang nantinya ketika dimasukkan persamaan akan menghasilkan DED yang berbeda ketika menggunkaan batasan sebelumnya akibat pengaruh perubahan batasan akibat *ramp-rate*.

Banyak metode yang digunakan dalam melakukan perhitungan ED. Salah satu metode yang digunakan pada Tugas Akhir ini adalah dengan menggunkan metode iterasi lambda [3]. Iterasi lambda sendiri memperhitungkan perubahan setiap lambda yang akan dimasukkan kedalam persamaan DED pada Tugas Akhir ini. Dikarenakan DED adalah jumlah perhitungan ED pada tiap waktu, maka pada Tugas Akhir ini kan didapatkan nilai lambda yang berfariasi tiap perubahan beban.

Simulasi yang digunakan pada Tugas Akhir ini diusulkan menggunakan Delphi. Delphi merupakan bahasa pemrograman berbasis Windows yang menyediakan fasilitas pembuatan aplikasi visual. Delphi memberikan kemudahan dalam menggunakan kode program, kompilasi yang cepat, penggunaan file unit ganda untuk pemrograman modular, pengembangan perangkat lunak, pola desain yang menarik serta diperkuat

dengan bahasa pemrograman yang terstruktur dalam bahasa pemrograman *Object Pascal*. Delphi memiliki tampilan khusus yang didukung suatu lingkup kerja komponen Delphi untuk membangun suatu aplikasi dengan menggunakan *Visual Component Library* (VCL). Sebagian besar pengembang Delphi menuliskan dan mengkompilasi kode program dalam *Integrated Development Environment* (IDE) [4]. Sehinga pada Tugas Akhir ini akan menghasilkan sebuah aplikasi perhitungan yang menggunakan bahasa yang dapat dipahami oleh pengguna. Aplikasi penyelesaian DED dengan memperhatikan *ramp-rate* yang lebih mudah untuk berinteraksi dengan pengguna.

## 1.2 Tujuan Penelitian

Tujuan yang ingin dicapai pada Tugas Akhir ini adalah sebagai berikut:

1. Membuat aplikasi perhitungan *Dynamic Economic Dispatch* (DED) dengan menggunkan metode iterasi lambda berbasis Delphi.
2. Melihat pengaruh *ramp-rate* pada perhitungan *Dynamic Economic Dispatch*.

## 1.3 Permasalahan

Dalam Tugas Akhir ini diharapkan permasalahan dapat terselesaikan, yaitu:

1. *Sofware* perhitungan DED yang telah dikembangkan dapat membantu pengguna dalam melakukan perhitungan DED dengan memperhatikan *ramp-rate* yang diseleasikan dengan metode iterasi lambda.
2. Memberikan pemahaman akan pengaruh *ramp-rate* dalam perhitungan DED

## 1.4 Batasan Masalah

Batasan masalah yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

1. Mengembangkan *sofware* perhitungan dikerjakan dengan menggunakan Delphi.
2. Metode yang digunakan untuk menyelasiakan permasalahan DED adalah metode itersi lambda
3. Perhitungan DED hanya memperhatikan pengaruh batasan *ramp-rate*.

4. Data rugi-rugi didapatkan dalam bentuk *matrix*.
5. Data rugi-rugi diperuntukkan untuk semua perhitungan DED di setiap periode.
6. Periode beban diasumsikan menggunakan interval waktu 1 jam
7. Data beban dalam aplikasi perhitungan DED tidak melebihi data total kemampuan pembangkit
8. Semua pembangkit diasumsikan selalu dalam keadaan menyala

## 1.5 Metode Penelitian

Pada Tugas Akhir ini dilakukan penelitian tentang *Dynamic Economi Dispatch* dengan memperhatikan batasan *ramp-rate* yang diselsaikan dengan menggunkan motode iterasi lambda. Perhitungan akan dilakukan dengan menggunkan aplikasi perhitungan berbasis Delphi. Tahap pengerjaan Tugas Akhir ini adalah:

1. Studi Literatur

Studi literatur untuk mencari referensi bahan melalui buku, jurnal ilmiah (*paper*), dan *browsing* melalui internet yang berhubungan dengan judul Tugas Akhir ini. Referensi yang dicari mencangkup teori *Dynamic Economic Dispatch* yang melihat pengaruh dari *ramp-rate*, Metode iterasi lambda yang dapat diterap dalam penyelesaian *output* pembangkitan dalam persamaan DED.

2. Pengumpulan Data, Perhitungan dan

Data yang bersangkutan dalam permasalahn DED seperti batasan *maximum* pembangkit, batasan *minimum* pembangkit, nilai *up-rate*, nilai *down-rate*, data beban tiap periode dalam interval waktu 1 jam, *fuelcost* dan nilai koefisien dari persamaan pembangkitan. Berdasarkan formulasi yang telah diketahui dilakukan perhitungan terlebih dahulu untuk menunjang pemodelan dan simulasi.

3. Perencanaan dalam Pembuatan Program

Dari data perhitungan tersebut dibuatlah strukur perancangan logika berupa diagram alir untuk proses pembuatan program yang nantinya akan diimplemantasikan kedalam bahasa pemrograman Delphi.

4. Simulasi Analisa Data

Setelah pembuatan program telah selesai dan menjadi sebuah aplikasi perhitungan, simulasi akan dicoba dengan menggunkan data inputan yang telah tersedia. Simulasi bertujuan untuk melihat aplikasi perhitungan DED yang dibuat telah sesuai dengan teori

perhitungan yang ada. Dari penjalanan simulasi akan didapatkan data yang akan dianalisa kebenarannya. Analisa bertujuan untuk memastikan hasil perhitungan program telah sesuai dengan metode yang digunakan. Data *output* akan kemudian dilihat apakah sesuai dengan batasan DED yang telah ditentukan.

5. Penulisan Buku

Hasil penelitian yang telah dilakukan akann dilaporkan dalam sebuah buku laporan Tugas Akhir. Isi dari laporan berdasarkan kesimpulan dari analisis yang telah didapat beserta tahapan yang ada di dalamnya.

## 1.6 Sistematika Penulisan

Sistematika penulisan dalam Tugas Akhir ini terdiri atas lima bab, dengan uraian sebagai berikut:

BAB 1 : Pendahuluan

Bagian ini membahas dasar-dasar penyusunan Tugas Akhir ini meliputi latar belakang, permasalahan yang diangkat, tujuan yang diharapkan, batasan masalah, metodologi pembuatan Tugas Akhir, sistematika dan relevansi penyusunan laporan Tugas Akhir ini.

BAB 2 : Tinjauan Pustaka

Bagian ini membahas teori-teori penunjang yang melandasi Tugas Akhir ini, formulasi perhitungan *Economic Dispatch* (ED), perkembangan ED yang disebut dengan *Dynamic Economic Dispatch* (DED), pengaruh *ramp-rate*, serta metode iterasi lambda sebagai metode yang digunakan dalam Tugas Akhir ini.

BAB 3 : Desain dan Simulasi

Bagian ini berisi proses desain, pemodelan serta simulasi yang dikerjakan agar didapatkan hasil aplikasi perhitungan *Dynamic Economic Dispatch* dengan memperhatikan *ramp-rate*, dimulai dengan cara melakukan perhitungan pada DED, kemudian dilanjutkan dengan proses penggunaan aplikasi perhitungan.

BAB 4 : Hasil Simulasi dan Analisis Data

Bagian ini membahas mengenai hasil simulasi yang didapatkan dari proses penyelesaian perhitungan yang dilakukan oleh aplikasi perhitungan yang telah dibuat,

serta analisa pengaruh *ramp-rate* pada perhitunga DED yang akan diuji dalam Tugas akhir ini.

BAB 5 : Penutup

Bagian ini membahas kesimpulan yang dapat diambil dari hasil penjalanan simulasi yang telah dilakukan analisa. Selain itu juga dilampirkan saran yang diharapkan mampu memberikan perbaikan serta penyempurnaan terkait keberlanjutan Tugas Akhir ini.

## 1.7 Relevansi

Hasil yang diperoleh dari Tugas Akhir ini diharapkan dapat memberikan kontribusi sebagai berikut

1. Dapat memberikan manfaat dalam kemudahan melakukan perhitungan *Dynamic Economic Dispatch* menggunakan iterasi lambda dengan memperhatikan *ramp-rate*.
2. Dapat menambah penguasaan ilmu dan teknologi di bidang optimalisasi pembangkitan sistem tenaga listrik.
3. Dapat menjadi reverensi Tugas Akhir untuk mahasiswa yang akan mengambil Tugas Akhir untuk mengembangkan aplikasi perhitungan DED dengan permasalahan yang berbeda.

*Halaman Ini Sengaja Dikosongkan*

# BAB 2
# DYNAMIC ECONOMIC DISPATCH

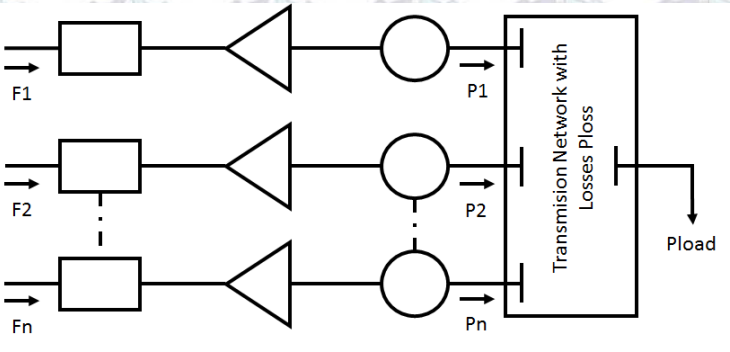
## 2.1 Sistem Tenaga Listrik

Sudah tidak dapat dipungkiri bahwa energi menempati peringkat yang sangat penting sebagai kebutuhan umat manusia. Sejak berabad-abad yang lalu setiap individu, kelompok maupun negara berjuang untuk memenuhi kebutuhannya akan energi. Hal tersebut mengakibatkan energi semakin langka dan harganya meningkat terus.

Salah satu bentuk energi yang sangat mudah dimanfaatkan adalah listrik. Pada era modern saat ini energi listrik ini menjadi factor penting bagi kehidupan manusia. Peralatan elektronik yang semakin berkembang menuntut manusia mengkonsumsi energi listrik untuk menunjang kebutuhan hidup sehari. Begitu juga dengn perusahaan modern saat ini yang menggunakan teknologi yang energi pengeraknya sendiri dari tenaga listrik.

Dilakukan berbagai kegitan untuk mengubah berbagai bentuk energi tersebut menjadi listrik dan kemudian sampai ke tangan konsumen. Proses tersebut dilakukan dalam sebuah kegiatan yang disebut dengan sistem tenaga listrik.

Sistem Tenaga Listrik adalah sistem penyediaan tenaga listrik yang terdiri dari beberapa pembangkit atau pusat listrik terhubung satu dengan lainnya oleh jaringan transmisi dengan pusat beban atau jaringan distribusi. Secara umum sistem tenaga listrik digambarkan dalam Gambar 2.1




**Gambar 2.1** N thermal unit mensuplai beban melalui jaringan transmisi

Pembangkit-pembangkit tenaga listrik dengan lokasi berjauhan satu sama lain terhubung pada sistem melalui sistem transmisi yang luas untuk mencatu tenaga listrik pada beban yang besar, disebut dengan sistem interkoneksi. Sistem interkoneksi menyebabkan:

1. Keandalan sistem yang semakin tinggi
2. Efisiensi pembangkittan tenaga listik dalam sistem meningkat
3. Mempermudah pejadwalan pembangkit

Kondidisi kesetimbangan antara pendapatan (penjualan) dan pengeluaran (pembiayaan) harus dijaga, agar dapat diperoleh keuntungan, sehingga kelangsungan unit usaha listrik dalam peruasahaan dapat terjaga.

Saat ini sistem tenaga mimiliki sistem intekoneksi *network* yang komplek [5]. Sitem tenaga dapat terdefinisikan bedasarkan empat bagian utama yang akan dijelaskan oleh subbab 2.1 berikut:

### 2.1.1 Generator

Salah satu bagian terpenting dari sistem tenaga listrik adalah pembangkit tenaga listrik. Stasiun pembangkit itu sendiri dapat berupa generator yang digerakkan dengan tenaga gas, tenaga air, tenaga disel dan lain sebagainya. Baiasanya jenis generator yang digunakan adalah generator 3 fasa atau generator sinkron. Sistem saat ini menggunakan generator AC dengan *rotating rectifier*, dikenal dengan sistem eksitasi *brushless*. Eksitasi generator bertujuan untuk menjaga tegangan pada generator dan mengontrol aliran daya reaktif. Akibat kekurangan komutator, generator AC dapat menghasilkan daya besar dengan tegangan tinggi sebesar 30 kV. Dalam *power plant*, ukuran daya yang dihasilkan berkisar antara 50 MW sampai 1500 MW.

Sumber tenaga mekanik biasa disebut dengan *prime mover* yang dapat berupa turbin *hydrolic* yang menggunakan tenaga potensial air terjun, turbin uap yang menggunakan sumber energi hasil pembakaran batu bara, dan lain- lain. Sumber-sumber energi alam dirubah oleh penggerak mula menjadi energi mekanis yang berupa kecepatan atau putaran, selanjutnya energi mekanis tersbut dirubah menjadi energi listrik oleh generator. Biasanya proses pembangkitan listrik oleh generator dihasilkan dengan menggunakan induksi elektromagnetik. Meskipun terdapat banyak kesamaan, generator berbeda dengan motor. Hal ini dikarenakan motor merupakan alat yang mengubah energi listrik menjadi energi mekanik. Sehingga dapat dikataka motor adalah beban yang justru membutuhkan energi listrik untuk mengerakkan rotor di dalamnya.

### 2.1.2 Transmisi dan Sub-Tranmisi

Transmisi listrik bertujuan untuk menyalurkan energi listrik dari unit pembangkit yang berasal dari berbagai tempat menuju sistem distribusi yang terhubung sebagai suplai kepada beban. Saluran transmisi juga menghubungkan peralatan selama sistem bekerja normal, maupun ketika terjadi gangguan.

Standart tegangan transmisi ditetapkan berdasarkan standart *American National Standart Institute* (ANSI). Tegangan transmisi bekerja pada jaringan lebih dari 60 kV distandarisasi menjadi 69 kv, 115 kv, 138 kv, 161 kv, 230 kv, 345 kv, 500 kv, dan 765 kv *line to line*. Tegangan tranmisi di atas 230 kv biasa disebut dengan tegangan ekstra tinggi.

Pada sistem transmisi yang terhubung dengan subtansi tegangan tinggi dilewatkan transformator *step-down* yang menuju subtansi distribusi. Beberapa industri besar memungkinkan untuk mendapatkan suplai langsung dari sistem sub-transmisi. Kapasitor bank dan reaktor bank biasanya terpasang pada subtansi untuk mempertahankan tegangan.

### 2.1.3 Distribusi

Sistem Distribusi merupakan bagian yang menghubungkan gardu induk distribusi pada konsumen sebagai masukan awal suplai tenaga terhadap beban. Besar tegangan saluran distribusi primer biasanya berkisar antaray 4 KV sampai dengan 34.5 KV dan mensuplai beban dalam area yang telah ditentukan.

Jaringan distribusi sekunder bertujuan untuk mengurangi tegangan yang diperuntukkan untuk penggunaan konsusmsi beban baik comersial maupun perumahan. Panjang kabel dipasang tidak melebihi 100 kaki yang terhubung dengan setiap konsumen. Distribusi sekunder kebanyakan melayani kosumen dengan level tegangan 240/120 V *single phase* tiga belitan, 280Y/12O V tiga *phase* empat belitan, atau 480Y/277 V tiga *phase* empat belitan.

Berdasarkan letak, sistem disribusi dibagi menjadi 2, yaitu:

1. *Overhead*, merupakan kabel atau kawat transmisi listrik disalurkan di udara atau di atas tanah. Memiliki kelebihan kemudahan dalam pemeliharaan dan kemudahan dalam perlusan wilayah, akan tetapi mudah mengalami gangguan.
2. *Underground*, merupakan kabel atau transimis listrik yang terletak di bawah tanah. Memiliki kelebihan tidak mudah mendapatkan gangguan, akan tetapi biaya pembangunan mahal

dan pemeliharaan lebih sulit karena letak yang berada di dalam tanah.

### 2.1.4 Beban Sistem

Daya beban real tercantum dalam satuan kilowatt (KW) atau megawatt (MW) dengan *magnitude* beban yang berfariasi setiap harinya, daya diharuskan mampu memenuhi kebutuhan beban. Masalah akan terjadi apabila daya yang dikirim lebih besar daripada kebutuhan beban, maka akan mengakibatkan kerugian pada perusahaan listrik akibat pemborosan energi. Sedangkan jika daya yang dikirim lebih kecil daripada permintaan beban, maka akan terjadi *over load* yang dapat mengakibatkan pemadaman.

Perkiraan beban merupakan masalah yang sangat menentukan bagi perusahaan listrik baik dalam segi manajerial maupun segi operasional. Oleh karenanya perlu mendapat perhatian khusus untuk dapat membuat perkiraan beban yang sebaik mungkin. Berdasarkan jangka waktunya perkiraan beban dapat dibedakan menjadi:

1. Perkiraan beban jangka panjang (*long term*), merupakan perkiraan beban listrik untuk jangka waktu di atas satu tahun.
2. Perkiraan beban jangka menengah (*medium term*), merupakan perkiraan beban lisrik untuk jangka waktu antara satu bulan sampai dengan satu tahun.
3. Perkiraan beban jangka pendek (*short term*), merupkan perkiraan beban listrik untuk jangka waktu beberapa jam dalam sehari sampai dengan satu minggu.

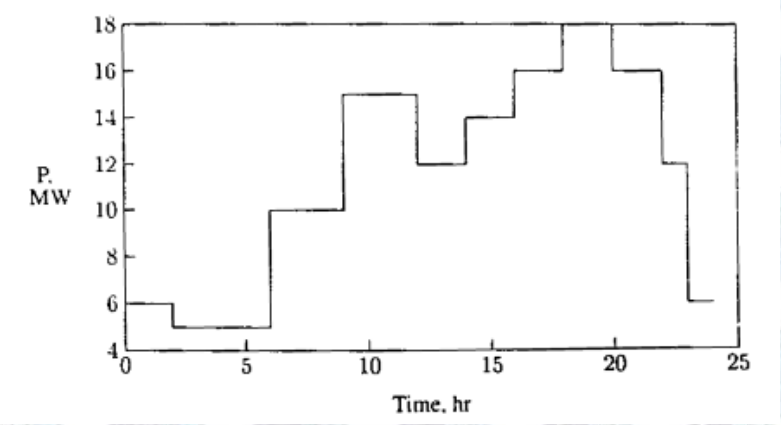
Berdasarkan daerah beban dapat dibagi menjadi:

1. Beban Industri, industri yang besar memungkinkan untuk mendapatkkan suplai daya dari jaringan subtransmisi dan industri kecil mendapatkan pelayanan daya dari jaringan distribusi primer. Beban industri merupakan beban gabungan yang memiliki fungsi tegangan dan frekuensi
2. Beban Komersial dan perumahan, beban jenis ini memiliki frekuensi yang relaltif tetap dan konsumsi daya reaktif yang kecil diabaikan.

Masing-masing sektor tersebut memiliki karakteristik beban yang berbeda, dikarenakan tiap konsumen disetiap sektor memiliki pola konsumsi yang berbeda.

Kurva beban secara sederhana dapat diartikan sebagai kurva yang menggambarkan penggunaan beban (listrik) dalam satuan waktu, entah

itu dalam selang waktu hari, minggu, atau tahun. Akan tetapi penggunaan kurva yang paling umum adalah kurva beban harian.




**Gambar 2.2** Kurva beban harian [4]

## 2.2 Karakteristik Pembangkit Thermal

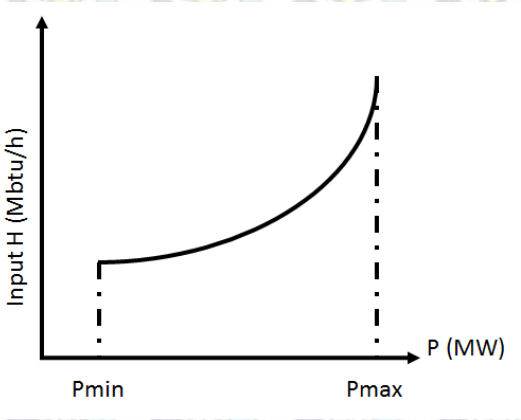
Ada banyak parameter dalam analisis pengaturan operasi sistem tenaga. Hal yang paling mendasar dalam masalah operasi ekonomis adalah karakteristik *input-output* pada pembangkit *thermal*. Untuk menggambarkan karakteristik *input-output*, *input* merepresentasikan sebagai masukan total yang diukur dalam satuan biaya/jam dan *output* merupakan daya keluaran listrik yang disediakan oleh sistem pembangkit tenaga listrik.Dalam menggambarkan karakteristik unit turbin uap, akan menggunakan *termionologi* (2.1) dan (2.2) sebagai berikut:

$$H = \frac{Mbtu}{jam} \tag{2.1}$$

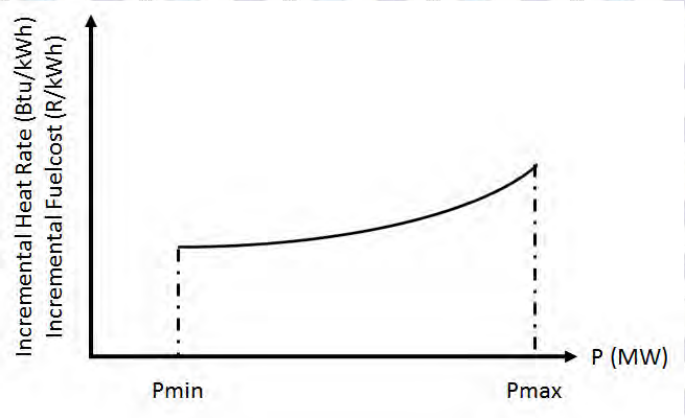
$$F = \frac{R}{jam} \tag{2.2}$$

H dapat dinyatakan sebagai energi panas yang dibutuhkan tiap jam dan F dinyatakan sebagai biaya tiap jam. Ada kalanya R/jam biaya operasional suatu unit terdiri dari biaya operasional dan biaya pemeliharaan. Biaya karyawan akan dimasukkan sebagai bagian dari biaya operasi jika biaya ini dapat digambarkan secara langsung sebagai fungsi dari *output* unit. *Output* dari unit pembangkit dinyatakan dengan P

dalam Megawatt. Untuk lebih jelasnya dapat dilihat pada Gambar 2.3 dan Gambar 2.4




**Gambar 2.3** Kurva input-output pembangkit thermal




**Gambar 2.4** Kurva incremental pembangkit thermal

Karakteristik *input-output* dari unit pembangkit *thermal* yang ideal, digambarkan sebagai kurva nonlinear yang kontinyu. Data karakteristik input output diperoleh dari perhitungan desain atau. Pembangkit *thermal* mempunyai batas operasi *minimum* (Pmin) dan *maxsimum* (Pmax). Batasan beban minimum biasanya disebabkan oleh kestabilan

pembakaran dan masalah desain generator. Pada umumnya unit pembangkit thermal tidak dapat beroperasi dibawah 30% dari kapasitas desain.

## 2.3 Economic Dispatch (ED)

Tingkat efisiensi dalam operasi optimalisasi ekonomi dan perencanaan daya pembangkitan listrik akan selalu menjadi bagian penting dalam perindustrian listrik. Oleh karena itu diperlukan perhitungan khusus akan pengiriman daya kepada para konsumen tenaga listrik sehinnga perusahaan pemasok listrik tidak mengalami kerugian.

Tujuan utama dari *Economic Dispatch* (ED) adalah untuk menentukan kombinasi daya *output* yang minimal dari setiap unit pembangkit, dengan meminimalkan total biaya bahan bakar, sementara dapat memenuhi kebutuhan baban para konsumen. Pengoptimalan permasalahan ED sangat penting untuk melakukan perkiraan jangka panjang dalam sistem tenaga listrik, penentuan porsi biaya, dan pemodelan managemen operasi tenaga listrik pada pembangkit.

Pembangkitan listrik memiliki tiga komponen biaya utama, antara lain biaya pembangunan, biaya kepemilikan, biaya operasionla. Biaya operasional merupakan biaya yang berkaitan langsung dengan keuntungan penjalanan produksi. Hal ini dikarenakan biaya operasional berhubungan langsung dengan managemen pembangkitan daya liatrik.

Salah satu bagian yang paling penting dalam biaya operasional adalah biaya bahan bakar (*fuelcost*). Pada setiap unit pembangkitan nilai yang berbeda tergantung dari jenis bahan bakar yang digunakan dalam pembangkitan. Nilai dari *fuelcost* sangat mempengaruhi fungsi biaya yang didapat. Secara umum nilai dari *fuelcost* dapat dinyatakan dalam persamaan (2.3) berikut.

$$fuelcost = \frac{R}{Mbtu} \tag{2.3}$$

*Fuelcost* merupakan harga persatuan panas dari bahan bakar, atau dapat dinyatakan sebagai konversi satuan panas ke satuan mata uang.

Pengaruh nilai *fuelcost* terhadap fungsi biaya dalam dilihat dalam persaman objektif ED berikut,

$$Hi(Pi) = aiPi^2 + biPi + ci \tag{2.4}$$

$$Fi(Pi) = Hi(Pi) \times fuelcost\ i \tag{2.5}$$

$$Ftotal = min \sum_{i=1}^{n} Fi(Pi) \quad (2.6)$$

Dimana

$n$ : jumlah generator

Dengan terhubungnya banyak unit pembangkit dalam sebuah sistem interkoneksi memberikan kemungkinan pengaturan pembangkitan yang lebih kecil untuk setiap unit.

*Equality Constrain* merupakan batasan yang merepresentasikan keseimbangan daya dalam sistem. Fungsi persamaan pada ED dinyatakan dalam persamaan,

$$\sum_{i=1}^{n} Pi = Pload + Ploss, n = jumlah\ generator \quad (2.7)$$

*Inequality Constrain* merupakan batasan yang merepresentasikan kapasitas daya dari pembangkit. Pada ED fungsi pertidaksamaan dinyatakan dalam persamaan (2.8) berikut.

$$Pi\ min \leq Pi \leq Pi\ max \quad (2.8)$$

Jika batasan *minimum* memiliki nilai seperti yang didapatkan pada persamaan (2.9) maka akan didapatkan solusi (2.10).

$$Pi \leq Pi\ min \quad (2.9)$$

$$Pi = Pi\ min \quad (2.10)$$

Jika batasan *maximum* memiliki nilai seperti yang didapatkan pada persamaan (2.11) maka akan didapatkan solusi (2.12).

$$Pi \geq Pi\ max \quad (2.11)$$

$$Pi = Pi\ max \quad (2.12)$$

## 2.4 Dynamic Economic Dispatch (DED)

*Dynamic Economic Dispatch* merupakan *Economic Dispatch* yang diperhitungkan dalam keadaan beban secara real time yang terus menerus berubah. Bentuk dari DED dapat dilihat pada persamaan objektif :

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci \quad (2.13)$$

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i \quad (2.14)$$

$$Ftotal = min \sum_{t=1}^{T} \sum_{i=1}^{n} Fi(Pi(t)) \quad (2.15)$$

Dimana
$n$ : jumlah generator
$T$ : total waktu dalam jam

*Equality constrain*

$$\sum_{i=1}^{T} Pi(t) = Pload,(t) + Ploss(t) \quad (2.16)$$

*Inequality constrain*

$$Pi\ min \leq Pi(t) \leq Pi\ max \quad (2.17)$$

## 2.5 Ramp-rate

Permintaan konsumsi listrik terus berubah, membuat variasi dan ketidakpastian karakteristik yang melekat pada sistem listrik. Akan tetapi generator sebagai penghasil listrik memiliki batasan tersendiri untuk menghasilkan daya yang optimal sesuai dengan perubahan beban. Dalam hal ini *ramp-rate* perlu diperhatikan sebagi batasan baru pembangkitan setiap unit generator.

*Ramp-rate* merupakan kemampuan generator untuk melakukan peningkatan (*up-rate*) atau penurunan (*down-rate*) generasi. Setiap unit pembangkit memiliki karakteristik yang berbeda, sehingga membutuhkan fungsi tertentu untuk mendapatkan hasil yang optimal.

Fungsi *ramp-rate* dapat dilihat dalam persamaan:

$$Pi(t-1) - Pi(t) \leq DRi \quad (2.18)$$

$$Pi(t) - Pi(t-1) \leq URi \quad (2.19)$$

Sehingga jika persamaan (2.18) dan (2.19) digabungkan akan mendapatkan batasan baru, yaitu:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1) \quad (2.20)$$

Nilai dari batasan *ramp-rate* tersebut tetap harus memenuhi batasan DED pada Tugas Akhir ini. Sehingga didapatkan syarat baru unuk perhitungan optimasi pada beban periode berikutnya.

Jika batasan *minimum ramp-rate* memiliki nilai yang didapatkan pada peramaan (2.21) maka akan didapatkan solusi (2.22).

$$Pi(t-1) - DRi < Pi\,min \quad (2.21)$$

$$Pi(t-1) - DRi = Pi\,min \quad (2.22)$$

Jika batasan *minimum ramp-rate* memiliki nilai yang didapatkan pada peramaan (2.23) maka akan didapatkan solusi (2.24)

$$URi + Pi(t-1) > Pi\,max \quad (2.23)$$

$$URi + Pi(t-1) = Pi\,max \quad (2.24)$$

## 2.6 Loss Formula

Dalam sistem tenaga, kerugian transmisi merupakan kehilangan daya yang harus ditanggung oleh unit pembangkitan. Sehingga daya yang hilang pada kerugian transmisi akan menjadi beban tambahan pada sistem tenaga.

Rugi-rugi dalam jaringan transmisi sistem tenaga akan mejadi sebuah fungsi pembangkitan. Berdasarkan formula rugi-rugi yang konstan, fungsi didapatkan dalam persamaan quadratik yang diselesaikan dengan mengetahui *B coefficient* [6], yang dapat dikenal sebagai *Bloss matrix*.

*Bloss matrix* formula sudah diperkenalkan sejal 1950 sebagai metode untuk mendapatkan nilai rugi-rugi dan *incremental loss* dalam perhitungan [7]. Dimana persamaan daya loss secara umum dapat dilihat pada persamaan:

$$Ploss = [P1 \quad \dots \quad Pn] \begin{bmatrix} B11 & \cdots & B1j \\ \vdots & \ddots & \vdots \\ Bi1 & \cdots & Bij \end{bmatrix} \begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + [Bo1 \quad \dots \quad Boi] \begin{bmatrix} P1 \\ \vdots \\ Pn \end{bmatrix} + Boo \quad (2.25)$$

Sehingga pada persamaan yang didapatkan pad (2.25) dapat disederhanakan menjadi persamaan (2.26) berikut:

$$Ploss = P^T[B]P + Bo^T P + Boo \quad (2.26)$$

Dimana,

$P$ : *vector* semua generator (MW)

$[B]$ : *matrix* persegi dari dimensi yang sama dengan P

$Bo$ : vector dengan panjang yang sama dengan P

$Boo$ : koefisien konstan

Nilai pada Bij secara umum merepresentasikan koefisien rugi-rugi dengan persamaan umum:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo \qquad (2.27)$$

Nilai dari rugi-rugi daya biasanya berkisar antara 20 % hingga 30 % dari jumlah semua total beban.

Ketika nilai rugo-rugi diperhitungkan maka *penalty factor* di setiap unit akan berbeda. Tidak seperti ketika mengitung nilai optimum pembangkitan tanpa menggunakan rugi-rugi, dimana nilai *penalty factor* dianggap 1 (satu). Persamaan *penalty factor* dapat dilihat pada persaman (2.28) berikut:

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}} \qquad (2.28)$$

Dimana memiliki hubungan dengan *incremental loss* yang dituliskan dalam persamaan (2.29) berikut:

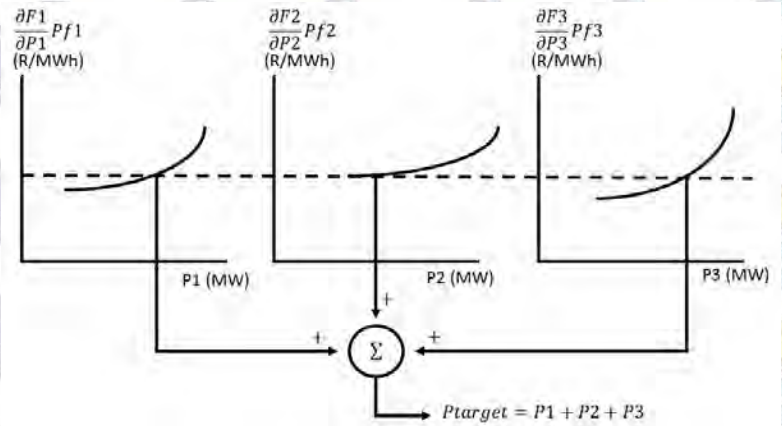
$$incremental\ loss = \frac{\partial Ploss}{\partial Pi} \qquad (2.29)$$

Nilai dari *penalty factor* akan mempengaruhi nilai lambda dalam perhitungan Tugas Akhir ini. Pengaruh dari *penalty factor* akan terlihat dalam persamaan yang akan dijelaskan dalam Subbab 2.7**.**

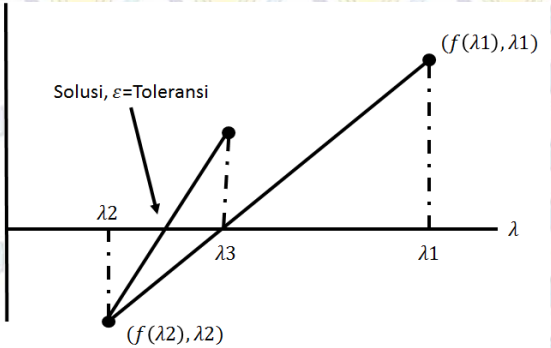
## 2.7 Iterasi Lambda

Iterasi Lambda digunakan pada Tugas Akhir ini untuk menyeleasaikan persamaan *Dynamic Economic Dispatch*.

Pada metode iterasi lambda, nilai lambda pertama akan ditentukan terlebih dahulu. Tentunya nilai dari lambda petama bukanlah hasil yang benar. Ketika niali total dari $P1 + P2 + P3 + \cdots Pi < Ptarget$ maka nilai $\lambda$ untuk itersi berikutnya akan bertambah lebih besar dari nilai $\lambda$ sebelumnya. Dan sebaliknya, jika nilaii total $P1 + P2 + P3 + \cdots Pi > Ptarget$ maka nilai lambda untuk iterasi berikutnya akan lebih kecil daripada nilai dari lambda sebelumnya. Proses ini akan melakukan itersi nilai lambda hingga mendapatkan hasil dimana $P1 + P2 + P3 + \cdots Pi = Ptarget$. Seperti yang dijelaskan dalam Gambar 2.5 dan Gambar 2.6




**Gambar 2.5** Grafik penyelesaian iterasi lambda




**Gambar 2.6** Proyeksi lambda

Nilai lambda dapat dilihat pada persamaan lagrangian:

$$\frac{\partial Fi}{\partial Pi} = \lambda \left(1 - \frac{\partial Ploss}{\partial Pi}\right) \tag{2.30}$$

$$\lambda = \frac{\partial Fi}{\partial Pi} \left(\frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}\right) \tag{2.31}$$

$$\lambda = \frac{\partial Fi}{\partial Pi} Pfi \quad (2.32)$$

Menentuka nilai lambda awal dilihat dalam persamaan 2.23:

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right) \quad (2.33)$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right) \quad (2.34)$$

$$\lambda start = \frac{\lambda \max + \lambda min}{2} \quad (2.35)$$

Untuk melakukan iterasi maka diperlukan $\Delta\lambda$, yang dapat dilihat dalam persamaan:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} \quad (2.36)$$

$$Ptarget = Ploss + Pload \quad (2.37)$$

Jika nilai yang didaptkan dalam proses iterasi sama dalam peersamaan (2.38), maka solusi peerubahan nilai lambda untuk iterasi berikutnya adalah (2.39).

$$\sum_{i}^{n} Pi - Ptarget > 0, \qquad n = Jumlah\ Generator \quad (2.38)$$

$$\lambda = \lambda sebelum - \Delta\lambda \quad (2.39)$$

Jika nilai yang didaptkan dalam proses iterasi sama dalam peersamaan (2.40), maka solusi peerubahan nilai lambda untuk iterasi berikutnya adalah (2.41).

$$\sum_{i}^{n} Pi - Ptarget < 0, \qquad n = Jumlah\ Generator \quad (2.40)$$

$$\lambda = \lambda sebelum + \Delta\lambda \quad (2.41)$$

Nilai lambda akan terus berubah hingga mendapatkan nilai dalam persamaan (2.42) berikut,

$$\sum_{i}^{n} Pi - Ptarget = 0, \qquad n = Jumlah\ Generator \quad (2.42)$$

Nilai $\lambda$ pada iterasi kedua biasanya bernilai 10 % lebih besar dari nilai $\lambda$ pertama, atau 10 % kurang dari nilai $\lambda$ pertama tergantug dari hasil perhitungan error pada iterasi tertentu.

Pada Tugas Akhir ini nilai *error* ($\varepsilon$) ditentukan sebesar 0.01. Ketika diimplementasikan ke dalam persamaan, maka nilai lambda akan berhenti melakukan iterasi hingga mendapatkan nilai,

$$\sum_{i}^{n} Pi - Ptarget = \varepsilon, \qquad n = Jumlah\ Generator \quad (2.43)$$

Dimana setiap nilai *Pi* harus memenuhi syarat *inequality constrain* dan batasan *ramp-rate* yang digunakan untuk penyelesaian masalah DED pada Tugas Akhir ini.

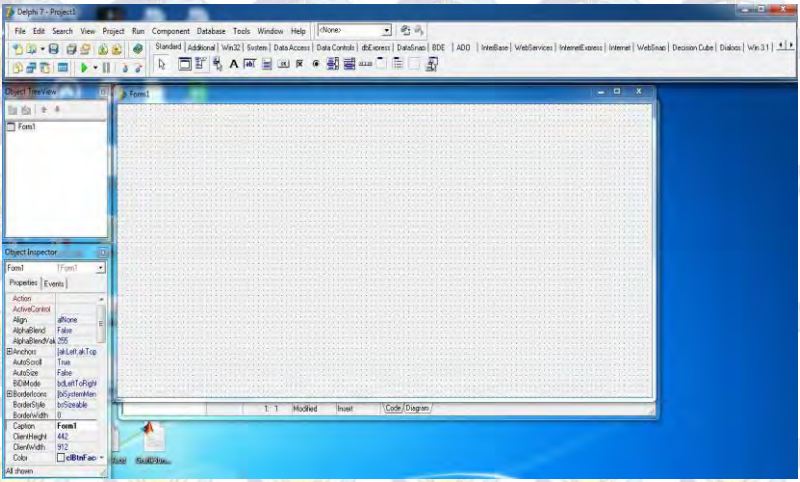
## 2.8 Delphi

Delphi merupakan bahasa pemrograman berbasis *Windows* yang menyediakan fasilitas pembuatan aplikasi *visual*. Pada Tugas Akhir ini menggunakan Delphi untuk mengaplikasikan formula DED yang ada. Dengan menghasilkan aplikasi perhitungan DED yang memiliki *interface* yang mudah dipahami, dengan batasan masalah yang telah ditentukan. Delphi sendirinya memiliki beberapa kelebihan diantaranya:

1. Kemudahan penyusunan *User Interface*, Delphi berkomitmen untuk menjadi *Rapid Aplication Development* (RAD). Maksudnya adalah bagaimana mempercepat perkembangan aplikasi.
2. Bahasa *Object Pascal*, merupakan salah satu varian dari bahas pascal dengan sejumlah penambahan, terutama terkait dengan dengan konsep *Object Oriented Programing* (OOP). Dengan salah satu kelebihan bahasa Pascal yang mudah dipahami dan tidak terlalu kompleks.
3. *Native Code*, hasil kompile Delphi adalah kode *native* untuk window 32. Ini berarti *file exe* yang dihasilkan oleh kompiler akan langsung dijalankan oleh mesin tanpa melalui *software* lain seperti *Virtual Machine* (VM). Secara umum *native code* lebih cepat daripada penjalan program dengan VM, hal ini dikarenakan Delphi menggunakan *installer* sederhana. Hasil dari Delphi adalah *file exe* tunggal, tanpa perlu file-file lainnya.

Ketika memulai Delphi, maka akan ditempatkan ke dalam *Integrated Development Environment* (IDE). IDE ini menyediakan semua

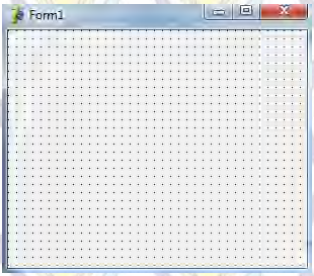
alat yang dibutuhkan dalam merancang, mengembangka, menguji, *debug*, dan penyebaran aplikasi dalam waktu yang singkat. Dalam hal ini IDE mencangkup semua alat yang diperlukan untuk memulai perancangan aplikasi seperti seperti yang terlihat pada Gambar 2.7 berikut.



**Gambar 2.7** Tampilan pengerjaan Delphi

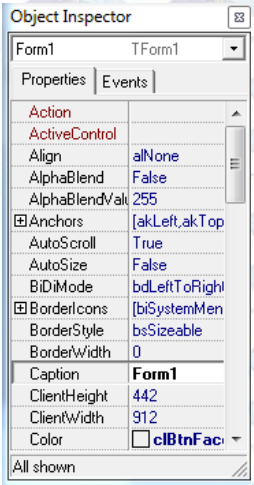
Beberapa bagian penting yang perlu diketahui dalam pemrograman dengan menggunakan Delphi diantaranya adalah:

1. *Form Designer* atau *Form*, merupakan jendela kosong yang digunakan untuk merancang suatu *User Interface* (UI) dalam perancangan aplikasi yang sedang dibuat. Dapat dilihat pada Gambar 2.8 berikut,



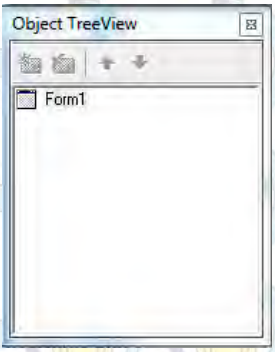

**Gambar 2.8** Tampilan *Form Designer*

2. *Object Inspector*, digunakan untuk mengatur dan memeriksa sekumpulan *property* yang berada di dalam *Form* untuk mendapatkan tampilan sesuai denganyang diinginkan. *Object Inspector* dapat dilihat pada Gambar 2.9.



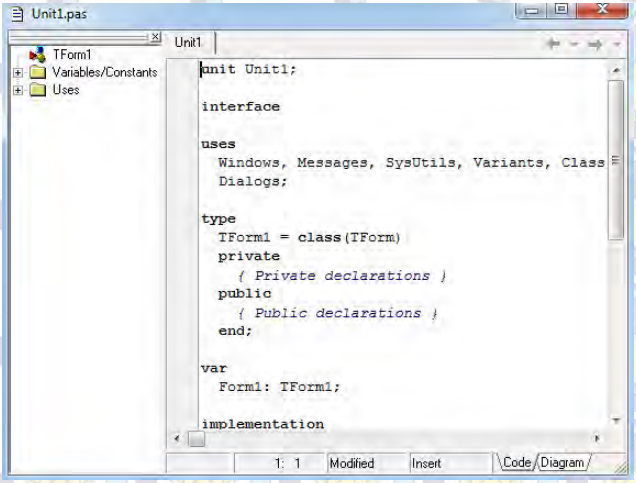

**Gambar 2.9** Tampilan *Object Inspector*

3. *Object TreeView*, digunkan untuk menampilkan dan mengubah hubungan antar komponen dalam *Form*. Tampilan dapat dilihat pada Gambar 2.10.




**Gambar 2.10** Tampilan *Object TreeView*

4. *Code Editor*, merupakan tempat penulisan dan edititing logika pemrograman yang sedang dibuat. Dapat dilihat pada Gambar 2.11.



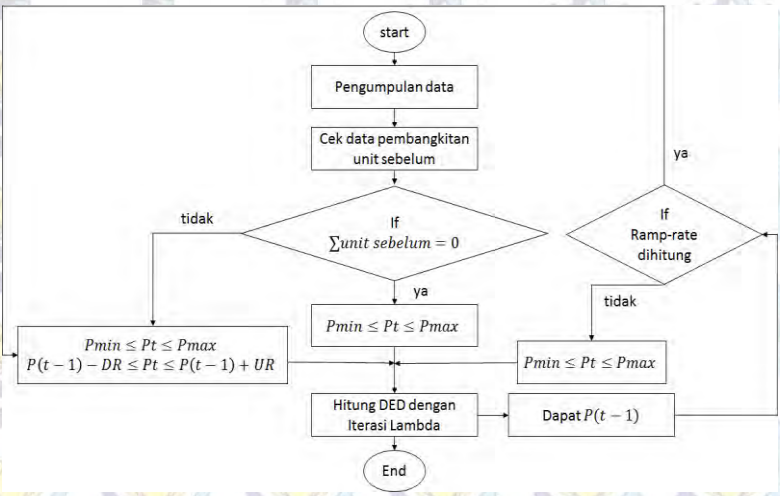

**Gambar 2.11** Tampilan *Code Editor*

*Halaman ini Sengaja Dikosongkan*

# BAB 3
# IMPLEMENTASI DYNAMIC ECONOMIC DISPATCH MENGGUNKAN ITERASI LAMBDA DENGAN MEMPERHATIKAN RAMP-RATE

Dalam Tugas Akhir ini, iterasi lambda digunakan untuk menyelesaikan permasalahan dalam perhitungan *Economic Dispatch* (ED) yang kemudian akan diterapkan pada *Dynamic Economic Dispatch* (DED). Hal ini dikarenakan perubahan batasan yang akan terjadi dalam persaaman perhitungan iterasi lambda pada DED. Pengaruh dari syarat batasan *ramp-rate* yang akan mempengaruhi perubahan batasan pembangkitan daya *maximum* dan daya *minimum* untuk mendapatkan hasil pembangkitan yang optimal. Sehingga dengan didapatkannya nilai kombinasi pembangkitan yang *minimum* akan didapatkan nilai biaya *minimum*. Pengolahan data dan simulasi menggunakan aplikasi perhitungan yang dibuat dengan menggunakan Delphi untuk membantu penyelesaian Tugas Akhir ini. Alur penyelesaian serta penerapan metode akan diterapkan pada subbab 3.1 berikut.

## 3.1 Alogaritma DED




**Gambar 3.1** Flowcart penerapan DED pada Delphi

Alur dari penggunaan aplikasi perhitungan DED dimulai dengan mengumpulkan semua data yang dibutuhkan. Mulai dari jumlah unit, koefisien tiap orde untuk *Incremental Heat Rate*, *fuelcost*, data batasan *maximum* dan *minimum* tiap unit, batasan *ramp-rate*, dan nilai pembangkitan unit pada jam sebelum. Setelah itu menentukan berapa beban yang ingin diperhitungkan dan mengisi data beban pada setiap jam. Perhitungan akan mengggunakan metode iterasi lambda untuk menentukan pembangkitan setiap unit. Hasil pembangkitan akan dimasukan ke dalam persamaan fungsi biaya untuk mendapatkan nilai biaya pada beban setiap jam yang telah ditentukan.

## 3.2 Inisiasi Persamaan Objektif dan Constrain DED

Seperti yang telah dijelaskan di dalam bab 2 sebelumnya, DED memiliki persamaan objektif yang dipergunkan untuk mendapatkan nilai pembangkitan dan biaya ketika dikalikan dengan *fuelcost*. Persamaan objektif DED dapat dilihat dalam persaman berikut:

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci \tag{3.1}$$

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i \tag{3.2}$$

Dan dengan persamaan *inequlity constrain* yang telah di pengaruhi oleh *ramp-rate* dapat dilihat dalam persamaan:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1) \tag{3.3}$$

Dalam program perhitungan Delphi juga diberikan persamaan rugi-rugi yang telah ditentukan nilainya berupa *Bloss matrix* yang akan diiterasikan hingga mencapai nilai rugi-rugi yang tidak berubah. Persamaan rugi-rugi dalam DED dapat dilihat dalam persamaan:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo \tag{3.4}$$

Sehingga dengan adanya penambahan nilai rugi-rugi akan menjadi beban tambahan dalam sistem operasi. Hal ini akan menghasilkan persamaan *equality constrain*, yaitu:

$$Ptarget = Ploss + Pload \tag{3.5}$$

$$\sum_{i}^{n} Pi - Ptarget = \varepsilon, \qquad n = Jumlah\ Generator \tag{3.6}$$

## 3.3 Agumen Input DED

Dari semua persaman dibuat argument inputan dalam bahasa Delphi sebagai masukan awal dalam perhitungan DED menggunakan iterasi lambda nantinya. Argumen yang digunakan dapat dilihat pada Tabel 3.1 sebagai berikut:

**Tabel 3.1** Argument input

| Argumen | Keterangan |
|---|---|
| Coeff[i] | Sebagai inputan awal dari nilai koefisien A, B, C dalam persamaan $Hi(Pi(t)) = aiPi(t)^2 + biPi(t) + ci$ |
| Unitmax[i] | Sebagai inputan awal dari batas *maximum* pembangkitan unit (Pmax) |
| Unitmin[i] | Sebagai inputan awal dari batasan *minimum* pembangititan unit (Pmin) |
| UR[i] | Sebagai inputan awal dari batasan atas *ramp-rate* (UR) |
| DR[i] | Sebagai inputan awal dari batasan bawah *ramp-rate* (DR) |
| Fuelcost[i] | Sebagai inputan awal nilai dari *fuelcost* yang digunakan untuk persaman $Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$ |
| Unitsebelum [i] | Sebagai inputan dan identifikasi hasil *Dynamic Economic Dispatch* pada jam sebelumnya. Hasil nilai yang didapatkan dari perhitungan akan digunakan untuk menentukan *inequality constrain* yang baru untuk perhitungan DED dengan beban pada periode berikutnya $Pi(t-1) - DRi \leq Pi(t)$ $Pi(t) \leq URi + Pi(t-1)$ |
| Loadjam[i] | Sebagai inputan nilai beban ($Pload$) di setiap periode |
| B00[i], B0[i], B[i,i], B[i,j] | Sebagai inputan dari nilai koefisien pada *Bloss matrix* dalam persamaan $Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo$ |

## 3.3 Sintaksis DED

Sintaksis program merupakan perintah yang digunakan untuk melakukan pemanggilan program dengan argument input yang telah kita tentukan. Sintaksis yang digunakan dalam Delphi dilihat dalam Tabel 3.2 sebagai berikut:

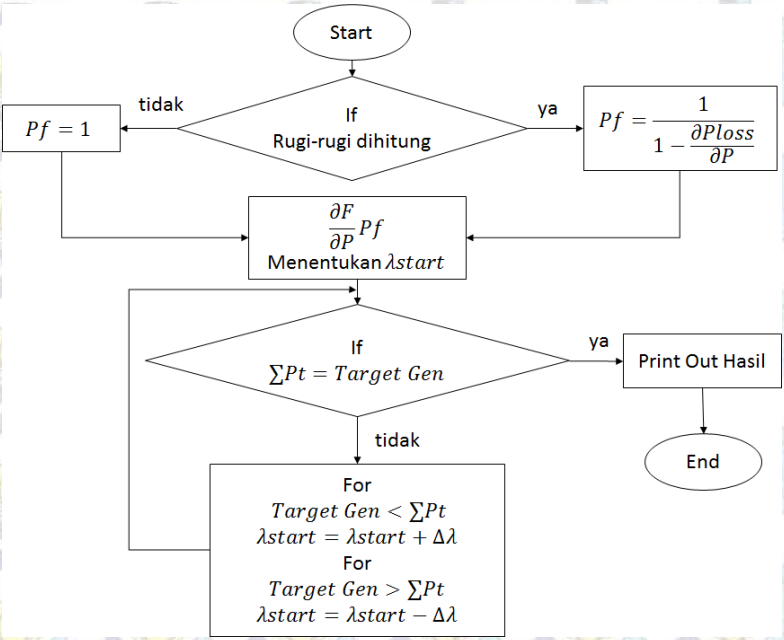
**Tabel 3.2** Sintaksis

| Sintaksis | Keterangan |
|---|---|
| datadump | Digunakan sebagai memasukkan semua data awal perhitungan di setiap periode |
| ihr_ftn | Sebagai inisiasi turunan pertama persamaan $Hi(Pi(t))$ yang nantinya akan dikalikan dengan *fuelcost*, sehingga didapatkan inisiasi $\frac{\partial Fi}{\partial Pi}$ |
| invers_ihr_ftn | Mencari nilai pembangkitan setiap unit ketika didapatkan nilai lambda $\frac{\partial Fi}{\partial Pi} = \lambda\left(1 - \frac{\partial Ploss}{\partial Pi}\right)$ |
| lambda_search_dispatch | Sebagai prosedure penjalanan metode iterasi lambda, yaitu dengan menentukan nilai $\lambda start$ dan $\Delta\lambda$ untuk proses iterasi. Hasil akhir bertujuan untuk mendapatkan jumlah nilai pembangkitan $\sum_{i}^{n} Pi - Ptarget = \varepsilon$ |
| loss_matrix_ftn | Mencari besaran nilai rugi-rugi dari persaman *Bloss matrix* |
| prod_cost | Mendapatkan nilai biaya pembangkitan setiap unit setelah mendapatkan nilai pembangkitan yang optimal dari proses iterasi lambda |

## 3.4 DED Meggunakan Metode Iterasi Lambda

Sebelum melakukan pemrograman yang kan diimplementasikan kedalam bahasa pemrograman Delphi yaitu bahasa *Pascal*, maka dibuat terlebih dahulu alur perhitungan DED.

Pada Tugas Akhir ini alur perhitungan DED menggunakan metode iterasi lambda, dari alur perhitungan didapatkan alur *flowcart* untuk dimplementasikan ke dalam Dlephi yang dapat digambarkan secara umum dalam Gambar 3.2 berikut,




**Gambar 3.2** Flowcart iterasi lambda

Pada proses iterasi dibutuhkan nilai lambda awal dengan menggunakan prsamaan (3.7) dan perubahan lambda untuk proses iterasi pada persamaan (3.8).

$$\lambda start = \frac{\lambda max - \lambda min}{2} \tag{3.7}$$

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} \tag{3.8}$$

Dimana nilai dari $\lambda max$ dapat ditentukan dengan menggunkan persamaan (2.34) dan nilai $\lambda min$ dapat ditentukan dengan menggunakan persamaan (2.33).

Beasarnya nilai lambda yang digunakan sebagai proses iterasi pada metode ini adalah,

$$\lambda = \frac{\Delta\lambda}{2} \tag{3.9}$$

Hingga didapatkan nilai lambda yang sesuai. Dalam hal ini nilai lambda akan berhenti ketika *equality constrain* telah terpenuhi. dengan syarat daya pembangkitan setiap unit harus memenuhi batasan,

$$Pi\ min \leq Pi(t) \leq Pi\ max \tag{3.10}$$

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1) \tag{3.11}$$

Dimana,

$$\frac{\partial Fi}{\partial Pi} = \lambda\left(1 - \frac{\partial Ploss}{\partial Pi}\right) \tag{3.12}$$

Diberikan contoh perhitungan untuk memahami alur dari penggunaan metode iterasi lambda pada DED. Dapat dilihat pada contoh kasus ini, melakukan perhitungan DED menggunakan sistem dengan 3 unit generator. Dengan permintaan daya setiap periode berintervalkan waktu 1 jam. Contoh ini akan menjelaskan hasil perhitungan DED pada load 2 jam pertama, dengan data sebagai berikut:

**Tabel 3.3** Contoh karakteristik pembangkit

| | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| A | 0.00142 | 0.00194 | 0.0048 |
| B | 7.2 | 7.85 | 7.97 |
| C | 510 | 310 | 78 |
| Pmin | 150 | 100 | 50 |
| Pmax | 600 | 400 | 200 |
| UR | 10 | 30 | 20 |
| DR | 10 | 30 | 20 |
| Fuelcost | 1.1 | 1 | 1 |

**Tabel 3.4** Contoh beban

| Jam | 1 | 2 |
|---|---|---|
| Beban | 850 | 800 |

Berikut adalah hasil perhitungan DED yang didapatkan ketika tidak melihat hasil rugi-rugi daya.

Pertama DED akan melakukan perhitunga untuk beban pada jam ke 1 (satu) kemudian melakukan perhitungan pada jam ke 2 (dua). Seperti data yag digunakan pada Tabel 3.2,

JAM 1, Beban 850 MW

Nilai *heat rate* :

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci$$

$$H1 = 510 + 7.2P1 + 0.00142P1^2$$

$$H2 = 310 + 7.85P2 + 0.00194P2^2$$

$$H3 = 78 + 7.97P3 + 0.00482P3^2$$

Fungsi biaya :

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$$

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$

Batasan tiap unit untuk perhitungan jam ke 1 mrnggunkan batasam maximum dan minimum yang dimilki oleh pembangkit, sebab asumsi untuk melakukan perhitungann DED pada jam ke 1 adalah dimana keadaan setiap unit pembangkit sedang dipersiapkan untuk menyala,maka batasan utuk perhitungan DED jam ke 1:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P_1 \leq 600$$

$$100 \leq P_2 \leq 400$$

$$50 \leq P_3 \leq 200$$

Menetukan lambda *minimum* dan lambda *maximum*, dikarenakan nilia rugi-rugi tidak diperhitungkan maka nilai $Pf = 1$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.2380$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.8980$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.0680$$

Menentukan perubahan lambda (dicari untuk melakukan iterasi):

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.83$$

Melakukan metode iterasi lambda,

Yaitu dengan memasukkan nilai lambda yang didapat kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load jam ke 1 yang diinginkan sebesar 850 MW.

Hasil proses iterasi lambda,

$$\lambda_{iter1} = 9.0680 \ldots\ldots\ldots totalgen = 795.3$$

$$\lambda_{iter2} = 9.8980 \ldots\ldots\ldots totalgen = 1200$$

$$\lambda_{iter3} = 9.4830 \ldots\ldots\ldots totalgen = 1057.3$$

$$\lambda_{iter4} = 9.2755 \ldots\ldots\ldots totalgen = 936.7$$

.
.

$$\lambda_{iter17} = 9.1483 \dots \dots \dots totalgen = 850.0$$

$$P1 = \frac{\lambda - 7.92}{2 \times 0.001562} = \frac{9.1483 - 7.92}{2 \times 0.001562} = 393.2$$
$$150 \leq P1 \leq 600$$
$$P1 = 393.2$$

$$P2 = \frac{\lambda - 7.85}{2 \times 0.00194} = \frac{9.1483 - 7.85}{2 \times 0.00194} = 334.6$$
$$100 \leq P2 \leq 400$$
$$P2 = 334.6$$

$$P3 = \frac{\lambda - 7.97}{2 \times 0.00482} = \frac{9.1483 - 7.97}{2 \times 0.00482} = 112.2$$
$$50 \leq P3 \leq 200$$
$$P3 = 112.2$$

Maka didapatkan hasil DED pembangkitan tiap unit untuk beban jam ke 1 sebesar 850 MW adalah,

$$P1 = 393.2$$

$$P2 = 334.6$$

$$P3 = 122.2$$

Nilai biaya operasi didapatkan setelah semua nilai pembangkitan setiap unit telah diketahui yang kemudian dimasukkan ke dalam fungsi biaya, maka didapatkan biaya operasi setiap unit sebagai berikut

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F1 = 561 + 7.92(383.2) + 0.001562(383.2^2)$$
$$F1 = 3825.06 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$
$$F2 = 301 + 7.85(306.1) + 0.00194(306.1^2)$$
$$F2 = 2894.52 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(110.7) + 0.00482(110.7^2)$$
$$F3 = 1019.78 \text{ (\$/jam)}$$

JAM 2, beban 800 MW
Batasan tiap unit jam ke 2:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P1 \leq 600$$

$$100 \leq P2 \leq 400$$

$$50 \leq P3 \leq 200$$

Syarat *ramp-rate*:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1)$$

Sehingga didapatkan batasan unit baru dengan memperhatikan kedua batasan.
Batasan baru tiap unit jam ke 2:

$$383.17 \leq P1 \leq 403.17$$

$$304.61 \leq P2 \leq 364.61$$

$$102.23 \leq P3 \leq 142.23$$

Menetukan lambda *minimum* dan lambda maximum, dikarenakan nilia rugi-rugi tidak diperhitungkan maka nilai $Pf = 1$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.9555$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pf, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.3411$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.0680$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.1928$$

Melakukan metode iterasi lambda,

Dengan memasukkan nilai lambda yang baru diakibatakan oleh pengaruh *ramp-rate* yang didapat kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 800 MW.

Hasil proses iterasi lambda,

$$\lambda_{iter1} = 9.1483 \dots \dots \dots totalgen = 850.0$$

$$\lambda_{iter2} = 8.9555 \dots \dots \dots totalgen = 790.0$$

$$\lambda_{iter3} = 9.0519 \dots \dots \dots totalgen = 805.2$$

$$\lambda_{iter4} = 9.0037 \dots \dots \dots totalgen = 795.0$$

$$\vdots$$

$$\lambda_{iter14} = 9.0376 \dots \dots \dots totalgen = 800.0$$

$$P1 = \frac{\lambda - 7.92}{2 \times 0.001562} = \frac{9.0376 - 7.92}{2 \times 0.001562} = 357.8$$

$$383.17 \leq P_1 \leq 403.17$$

$$P1 = 383.2$$

$$P2 = \frac{\lambda - 7.85}{2 \times 0.00194} = \frac{9.0376 - 7.85}{2 \times 0.00194} = 306.1$$
$$304.61 \leq P_2 \leq 364.61$$
$$P2 = 306.1$$

$$P3 = \frac{\lambda - 7.97}{2 \times 0.00482} = \frac{9.0376 - 7.97}{2 \times 0.00482} = 110.7$$
$$102.23 \leq P_3 \leq 142.23$$
$$P3 = 110.7$$

Maka didapatkan hasil DED pembangkitan tiap unit beban jam ke 2 sebesar 800 MW adalah,

$$P1 = 383.2$$

$$P2 = 306.1$$

$$P3 = 110.7$$

Nilai biaya operasi

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F1 = 561 + 7.92(383.2) + 0.001562(383.2^2)$$
$$F1 = 3825.06 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$
$$F2 = 301 + 7.85(306.1) + 0.00194(306.1^2)$$
$$F2 = 2894.52 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(110.7) + 0.00482(110.7^2)$$
$$F3 = 1019.78 \text{ (\$/jam)}$$

Ketika nilai rugi-rugi diperhitungkan, Hasil pembangkitan pada tiap periode akan berubah menjadi lebih besar dari sebelumnya. Hal ini dikarenakan rugi-rugi akan menjadi daya tambahan sebagai beban. Nilai lambda pada proses iterasi akan berubah dikarenakn pengaruh dari $Pf \neq 1$. Jika dibandingkan dengan proses iterasi yang tidak melihat rugi-rugi daya, dimana nilai $Pf = 1$.

Nilai Pf didapatkan dalam persamaan berikut,

$$Pf = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}} \tag{3.13}$$

Dengan adanya besaran nilai rugi-rugi daya akan mengakibatkan penambahan beban. Sehingga pembangkitan akan berubah menjadi lebih besar dikarenakan beban yang bertambah.

Diberikan contoh kasus dengan menggunkan data tiga unit pembnagkit yang telah diberikan pada Tabel 3.1 dan data beban pada Tabel 3.2. Persaman rugi- rugi dinyatakan dalam sebuah *Bloss matrix* yang telah ditentukan pada data berikut,

**Tabel 3.5** Contoh *Bloss matrix*

| Bij | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.00003 | 0 | 0 |
| 2 | 0 | 0.00009 | 0 |
| 3 | 0 | 0 | 0.00012 |

Berikut adalah perhitungan DED yang didapatkan dengan memperhatikan nilai rugi-rugi daya yang didapatkan,

JAM 1, beban 850 MW

Nilai *heat rate* :

$$Hi\big(Pi(t)\big) = aiPi(t)^2 + biPi(t) + ci$$

$$H1 = 510 + 7.2P1 + 0.00142P1^2$$

$$H2 = 310 + 7.85P2 + 0.00194P2^2$$

$$H3 = 78 + 7.97P3 + 0.00482P3^2$$

Fungsi biaya :

$$Fi(Pi(t)) = Hi(Pi(t)) \times fuelcost\ i$$

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$

Batasan tiap unit jam ke 1:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P1 \leq 600$$

$$100 \leq P2 \leq 400$$

$$50 \leq P3 \leq 200$$

Besarnya nilai rugi-rugi daya pada iterasi 1:

$$Ploss = \sum_{i}^{n}\sum_{j}^{n} Pi\ Bij\ Pj + \sum_{i}^{n} Bio\ Pi + Boo$$

$$Ploss = 0.00003P1^2 + 0.00009P2^2 + 0.00012P3^2$$

$$P1 = \frac{I50 + 600}{2} = 375$$

$$P2 = \frac{100 + 400}{2} = 250$$

$$P3 = \frac{50 + 200}{2} = 125$$

$$Ploss = 11.7$$

Menentukan besar beban baru akibat penamnhan nilai rugi-rugi pada iterasi 1,

$$Ptarget = Ploss + Pload$$
$$Ptarget = 11.7 + 850 = 861.7$$

Ketika nilai rugi-rugi diperhitungkan, penetukan nilai lambda min dan lambda max dipengaruhi oleh nilai $Pf$. Hal ini dikarenakan nilai dari $Pf \neq 1$, sehingga nilai lambda min dan max akan didapatkan,

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(375) = 0.0225$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(250) = 0.045$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(125) = 0.03$$

$$Pfi = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1 - 0.0225}$$

$$Pf2 = \frac{1}{1 - 0.045}$$

$$Pf3 = \frac{1}{1 - 0.03}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.5817$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 10.2041$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.3929$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.8112$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 1,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 861.7 MW.

$$\lambda_{iter1} = 9.3929 \dots\dots\dots totalgen = 810.92$$

$$\lambda_{iter2} = 10.2041 \dots\dots\dots totalgen = 1200$$

$$\lambda_{iter3} = 9.7985 \dots\dots\dots totalgen = 1078.48$$

$$\lambda_{iter4} = 9.5957 \dots\dots\dots totalgen = 944.7$$

.
.
.

$$\lambda_{iter15} = 9.4699 \dots\dots\dots totalgen = 861.7$$

$$P1 = \frac{\lambda(1 - 0.0225) - 7.92}{2 \times 0.001562} = \frac{9.4699(1 - 0.0225) - 7.92}{2 \times 0.001562}$$

$$150 \leq P1 \leq 600$$

$$P1 = 427.9$$

$$P2 = \frac{\lambda(1 - 0.045) - 7.85}{2 \times 0.00194} = \frac{9.4699(1 - 0.045) - 7.85}{2 \times 0.00194}$$

$$100 \leq P2 \leq 400$$

$$P2 = 307.67$$

$$P3 = \frac{\lambda(1-0.03)-7.97}{2\times 0.00482} = \frac{9.4699(1-0.03)-7.97}{2\times 0.00482}$$

$$50 \le P3 \le 200$$

$$P3 = 126.12$$

Nilai $P1, P2, P3$ yang didapatkan kemudian digunakan untuk mencari rugi-rugi iterasi 2

$$PLoss = 0.00003(427.9^2) + 0.00009(307.67^2) + 0.00012(126.12.8^2)$$

$$Ploss = 15.9$$

Menentukan besar beban baru akibat penamnhan nilia rugi-rugi pada iterasi 2,

$$Ptarget = Ploss + Pload$$

$$Ptarget = 15.9 + 850 = 865.9$$

Menentukan lambda *minimum* dan lambda *maximum*:

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(427.9) = 0.02685$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(307.67) = 0.068103$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(126.12) = 0.03356$$

$$Pfi = \frac{1}{1-\frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1-0.02685}$$

$$Pf2 = \frac{1}{1-0.068103}$$

$$Pf3 = \frac{1}{1 - 0.03356}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 8.6097$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 10.2070$$

Menentukan lambda start:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.4083$$

Menentukan delta lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.7986$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 2,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 865.9 MW.

$$\lambda_{iter1} = 9.4083 \ldots\ldots\ldots totalgen = 786.08$$

$$\lambda_{iter2} = 10.2070 \ldots\ldots\ldots totalgen = 1200$$

$$\lambda_{iter3} = 9.8076 \ldots\ldots\ldots totalgen = 1048.015$$
$$\lambda_{iter4} = 9.5081 \ldots\ldots\ldots totalgen = 851.566$$

.
.

$$\lambda_{iter16} = 9.5300 \dots \dots \dots totalgen = 865.9$$

$$P1 = \frac{\lambda(1 - 0.02567) - 7.92}{2 \times 0.001562} = \frac{9.53(1 - 0.02567) - 7.92}{2 \times 0.001562}$$

$$150 \leq P1 \leq 600$$

$$P1 = 437.1$$

$$P2 = \frac{\lambda(1 - 0.05539) - 7.85}{2 \times 0.00194} = \frac{9.53(1 - 0.05539) - 7.85}{2 \times 0.00194}$$

$$100 \leq P2 \leq 400$$

$$P2 = 278.9$$

$$P3 = \frac{9.53(1 - 0.03027) - 7.97}{2 \times 0.00482} = \frac{9.53(1 - 0.03027) - 7.97}{2 \times 0.00482}$$

$$50 \leq P3 \leq 200$$

$$P3 = 137.1$$

Nilai $P1, P2, P3$ yang didapatkan pada iterasi ke 2, kemudian digunkan untuk mencari rugi-rugi iterasi ke 3. Hal ini kan terus berlangsung hingga didapatkan nilai $P1, P2, P3$ pada rugi-rugi iterasi ke 11. Nilai rugi-rugi yang didapatkan pada iterasi ke 11 adalah sebesar 15.8 Sehingga total load yang diinginkan akan sebesar 865.8.

Didapatkan nilai lambda yang telah diiterasi untuk rugi-rugi pada iterasi ke 11 sebesar 9.5284, sehingga nilai yang pembangkitan setiap unit yang didapatkan sebesar

$$P1 = 435.2$$

$$P2 = 300.0$$

$$P3 = 130.7$$

Sehingga nilai biaya pada periode pertama didapatkan

$$F1 = 561 + 7.92P1 + 0.001562P1^2$$

$$F_1 = 561 + 7.92(435.2) + 0.001562(435.2^2)$$

$$F1 = 4303.59 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$

$$F2 = 301 + 7.85(300) + 0.00194(300^2)$$
$$F2 = 2839.31 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(130.7) + 0.00482(130.7^2)$$
$$F3 = 1201.65 \text{ (\$/jam)}$$

JAM 2, beban 800 MW

Batasan tiap unit jam ke 2:

$$Pi\ min \leq Pi(t) \leq Pi\ max$$

$$150 \leq P1 \leq 600$$

$$100 \leq P2 \leq 400$$

$$50 \leq P3 \leq 200$$

Syarat *ramp-rate*:

$$Pi(t-1) - DRi \leq Pi(t) \leq URi + Pi(t-1)$$

Sehingga didapatkan batasan unit baru dengan memperhatikan kedua batasan.

Batasan baru tiap unit jam ke 2:

$$425.20 \leq P1 \leq 445.20$$

$$269.97 \leq P2 \leq 329.97$$

$$110.66 \leq P3 \leq 150.66$$

Besarnya nilai rugi-rugi daya pada iterasi 1:

$$Ploss = 0.00003P1^2 + 0.00009P2^2 + 0.00012P3^2$$
$$P1 = \frac{425.2 + 445.2}{2} = 435.2$$

$$P2 = \frac{269.97 + 329.97}{2} = 299.97$$

$$P3 = \frac{110.66 + 150.66}{2} = 130.66$$

$$Ploss = 15.8$$

Menentukan besar beban baru akibat penambahan nilai rugi-rugi pada iterasi 1,

$$Ptarget = Ploss + Pload$$

$$Ptarget = 15.8 + 800 = 815.8$$

Menetukan lambda *minimum* dan lambda *maximum*:

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(435.2) = 0.02611$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(299.97) = 0.05399$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(130.66) = 0.03136$$

$$Pfi = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1 - 0.02611}$$

$$Pf2 = \frac{1}{1 - 0.05399}$$

$$Pf3 = \frac{1}{1 - 0.03136}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \dots Jumlah\ Generator\right)$$

$$\lambda min = 9.3293$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \dots Jumlah\ Generator\right)$$

$$\lambda max = 9.7274$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.5284$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.1990$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 1,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 815.8 MW.

$$\lambda_{iter1} = 9.5284 \dots \dots \dots totalgen = 865.83$$

$$\lambda_{iter2} = 9.3293 \dots \dots \dots totalgen = 805.824$$

$$\lambda_{iter3} = 9.4288 \dots \dots \dots totalgen = 821.56$$

$$\lambda_{iter4} = 9.3791 \dots \dots \dots totalgen = 810.83$$

.

.

.

$$\lambda_{iter13} = 9.4122 \dots \dots \dots totalgen = 815.8$$

$$P1 = \frac{\lambda(1 - 0.02611) - 7.92}{2 \times 0.001562} = \frac{9.4122(1 - 0.02611) - 7.92}{2 \times 0.001562}$$
$$425.2 \leq P1 \leq 445.2$$
$$P1 = 425.2$$

$$P2 = \frac{\lambda(1 - 0.05399) - 7.85}{2 \times 0.00194} = \frac{9.4122(1 - 0.05399) - 7.85}{2 \times 0.00194}$$
$$269.97 \leq P2 \leq 329.97$$
$$P2 = 271.6$$

$$P3 = \frac{\lambda(1 - 0.03136) - 7.97}{2 \times 0.00482} = \frac{9.4122(1 - 0.03136) - 7.97}{2 \times 0.00482}$$
$$110.66 \leq P3 \leq 150.66$$
$$P3 = 118.9$$

Nilai $P1, P2, P3$ yang didapatkan kemudian digunakan untuk mencari rugi-rugi iterasi 2

$$PLoss = 0.00003(425.2^2) + 0.00009(271.6^2) + 0.00012(118.9^2)$$

$$Ploss = 13.8$$

Menentukan besar beban baru akibat penambahan nilia rugi-rugi pada iterasi 2,

$$Ptarget = Ploss + Pload$$

$$Ptarget = 13.8 + 800 = 813.8$$

Menetukan lambda *minimum* dan lambda *maximum*:

$$\frac{\partial Ploss}{\partial P1} = 2(0.00003)(425.2) = 0.025512$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00009)(271.6) = 0.048888$$

$$\frac{\partial Ploss}{\partial P1} = 2(0.00012)(118.9) = 0.028536$$

$$Pfi = \frac{1}{1 - \frac{\partial Ploss}{\partial Pi}}$$

$$Pf1 = \frac{1}{1 - 0.025512}$$

$$Pf2 = \frac{1}{1 - 0.048888}$$

$$Pf3 = \frac{1}{1 - 0.028536}$$

$$\lambda min = min\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda min = 9.3024$$

$$\lambda max = max\left(\frac{\partial Fi}{\partial Pi} Pfi, i = 1 \ldots Jumlah\ Generator\right)$$

$$\lambda max = 9.6993$$

Menentukan lambda awal:

$$\lambda start = \frac{\lambda max + \lambda \min}{2} = 9.5009$$

Menentukan perubahan lambda:

$$\Delta\lambda = \frac{\lambda max - \lambda min}{2} = 0.1985$$

Melakukan metode iterasi lambda untuk rugi-rugi iterasi 2,

Dengan memasukkan nilai lambda yang baru, kemudian melakukan iterasi pada lambda dengan cara menambah dan mengurangi nilai lambda sebesar $\frac{\Delta\lambda}{2}$. Hal ini bertujuan untuk mendapatkan hasil pembangkitan daya yang sesuai dengan load yang diinginkan. Dalam kasus ini load yang diinginkan sebesar 813.8 MW.

$$\lambda_{iter1} = 9.5009 \dots \dots \dots totalgen = 864.87$$

$$\lambda_{iter2} = 9.3024 \dots \dots \dots totalgen = 805.82$$

$$\lambda_{iter3} = 9.4016 \dots \dots \dots totalgen = 827.28$$

$$\lambda_{iter4} = 9.3520 \dots \dots \dots totalgen = 810.82$$

.

.

.

$$\lambda_{iter13} = 9.3626 \dots \dots \dots totalgen = 813.8$$

$$P1 = \frac{\lambda(1 - 0.0255) - 7.92}{2 \times 0.001562} = \frac{9.3626(1 - 0.0255) - 7.92}{2 \times 0.001562}$$

$$425.2 \leq P1 \leq 435.2$$

$$P1 = 425.2$$

$$P2 = \frac{\lambda(1 - 0.0489) - 7.85}{2 \times 0.00194} = \frac{9.3626(1 - 0.0489) - 7.85}{2 \times 0.00194}$$

$$269.97 \leq P2 \leq 329.97$$

$$P2 = 271.88$$

$$P3 = \frac{\lambda(1 - 0.02854) - 7.97}{2 \times 0.00482} = \frac{9.3626(1 - 0.02854) - 7.97}{2 \times 0.00482}$$

$$110.66 \leq P3 \leq 150.66$$

$$P3 = 116.75$$

Nilai $P1, P2, P3$ yang didapatkan pada iterasi ke 2, kemudian digunkan untuk mencari rugi-rugi iterasi ke 3. Hal ini kan terus berlangsung hingga didapatkan nilai $P1, P2, P3$ pada rugi-rugi iterasi ke 11. Nilai rugi-rugi yang didapatkan pada iterasi ke 11 adalah sebesar 13.7 Sehingga total load yang diinginkan akan sebesar 813.7.

Didapatkan nilai lambda yang telah diiterasi untuk rugi-rugi pada iterasi ke 11 sebesar 9.3608, sehingga nilai yang pembangkitan setiap unit yang didapatkan sebesar,

$$P1 = 425.2$$

$$P2 = 271.5$$

$$P3 = 117.0$$

Sehingga nilai biaya pada beban jam ke 2 didapatkan

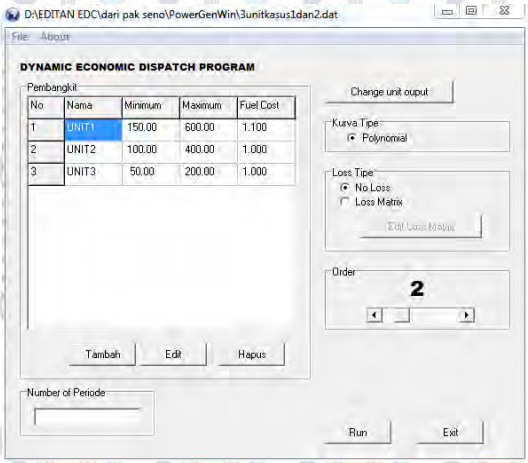
$$F1 = 561 + 7.92P1 + 0.001562P1^2$$
$$F1 = 561 + 7.92(425.2) + 0.001562(425.2^2)$$
$$F1 = 4210.95 \text{ (\$/jam)}$$

$$F2 = 301 + 7.85P2 + 0.00194P2^2$$
$$F2 = 301 + 7.85(271.5) + 0.00194(271.5^2)$$
$$F2 = 2584.18 \text{ (\$/jam)}$$

$$F3 = 78 + 7.97P3 + 0.00482P3^2$$
$$F3 = 78 + 7.97(117) + 0.00482(117^2)$$
$$F3 = 1076.54 \text{ (\$/jam)}$$

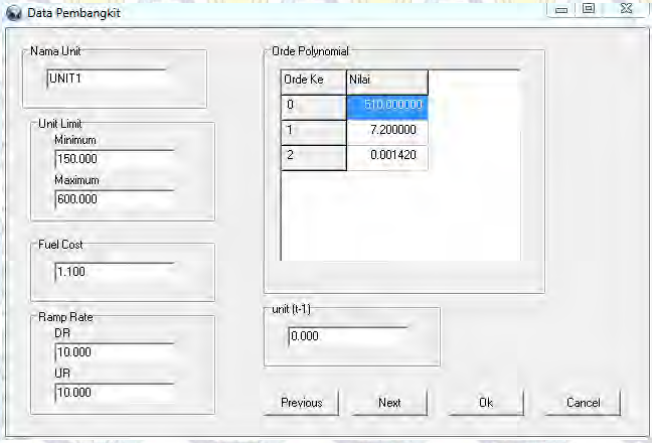
## 3.5 Aplikasi Perhitungan DED Iterasi Lambda

Berikut akan dijelaskan mengenai aplikasi perhitungan yang digunkan dalam Tugas Akhir ini. Seperti yang telah dinyatakan sebelumnya aplikasi perhitungan dibuat dengan menggunkan Delphi. Aplikasi yang digunakan khusus untuk melakukan perhitungan kasus penyelesaian DED. Berikut adalah tampilan aplikasi pada Gambar 3.3.




**Gambar 3.3** Tampilan utama aplikasi perhitugan DED

Tombol ‘Tambah’, ‘Edit’, dan ‘Hapus’ digunkan untuk mengisi data karakteristik pembagit yang telah ditentukan. Label kosong pada ‘Number of Periode’ diisi untuk menentukan jumlah periode beban yang ingin diperhitungkan DED. Ketika tombol ‘Edit’ digunkan maka akan muncul form pengisian data pembangkit seperti yang dipelihatkan pada Gambar 3.4.

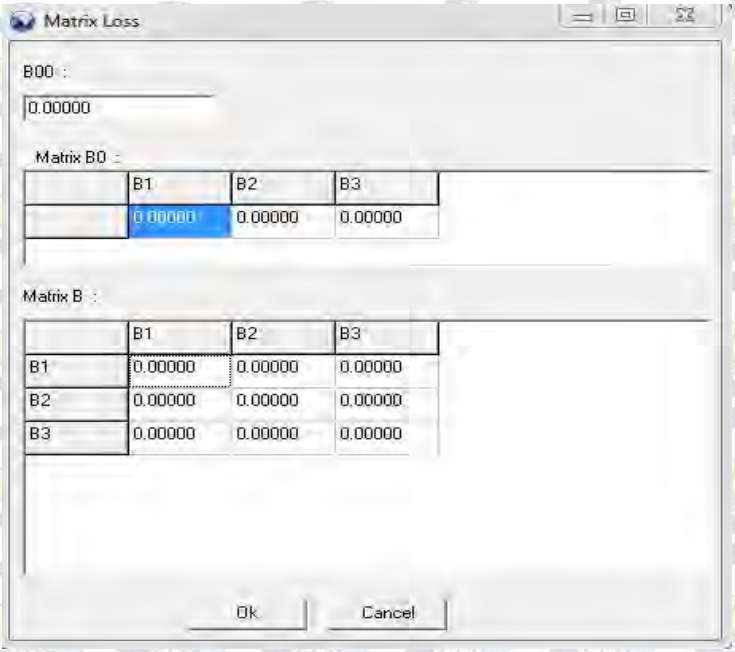



**Gambar 3.4** Tampilan pengisian karakteristik pembangkit

Gambar 3.4 memperlihatkan, bagian label ‘Minimum’ dan ‘Maximum, diisi dengan nilai batasan *minimum* dan *maximum* masing-masing unit. Label ‘Fuelcost’ diisi dengan nilai *fuelcost* setiap unit. Label ‘DR’ dan ‘UR’ berisikan nilai batsan *ramp-rate* setiap unit. Pada tabel ‘Orde Polynomial’ orde 0 berisi koefisen C, orde1 berisi koefisien B dan orde 2 berisi koefisien A pada persaman (2.13). Label ‘unit (t-1)’ diisikan nilai pembangkitan pada periode sebelumnya, jika diketahui.

Tombol ‘Next’ dan ‘Previous’ digunakan untuk memindah form pengisian data katarakteristik pembangkit sebelum dan sesudah. Tombol ‘Ok’ ditekan setelah data yang digunakan telah selesai diisi.

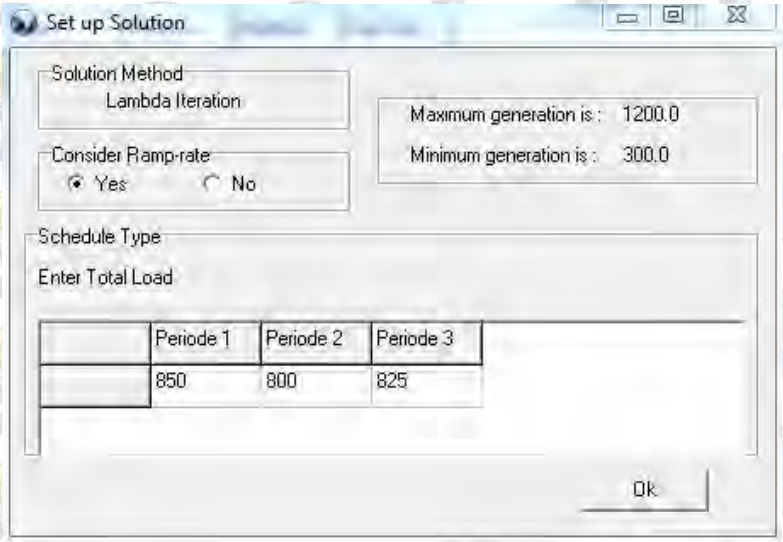
Pada Gambar 3.3, diberikan dua pilihan pengerjaan yaitu, ketika tidak melihat nilai rugi-rugi dengan memilih pilihan ‘No loss’, atau dengan memperhatikan nilia rugi-rugi dengan memilih ‘Loss Matrix’ yang kemudian dilanjutkan dengan menekan tombol ‘Edit Loss Marix’, maka akan keluar form pengisiaan data *Bloss matrix* seperti yang diperlihatkan pada Gambar 3.5,




**Gambar 3.5** Tampilan pengisian *Bloss matrix*

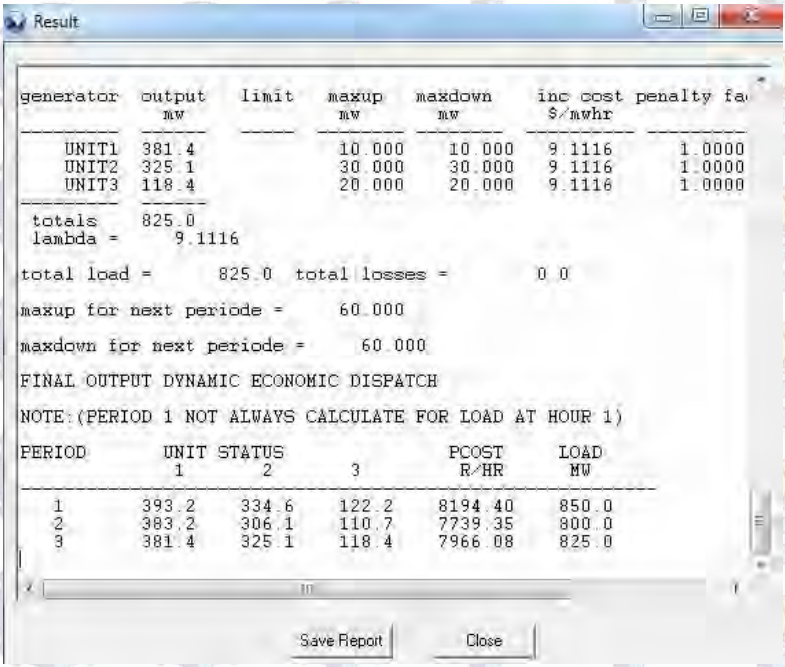
Pada Gambar 3.5 diberikan contoh ketika digunakan tiga unit pembangkitan. Sehingga kolom 'Matrix B0' akan berjumlah $1 \times 3$ dan 'Matrix B' akan berjumlah $3 \times 3$, sedangkan 'B00' akan selalu berisikan 1 kolom. Hal ini telah disesuikan dengan penggunaan rumus rugi-rugi yang konstan pada persamaan (2.27). Setelah semua bilangan yang dibutuhan telah terisi, maka tombol 'Ok' ditekan dan form akan kembali pada Gambar 3.3.

Setelah semua data yang digunakan telah diimputkan kemudian tekan tombol 'Run' pada Gambar 3.3, maka akan keluar form pengisian beban setiap periode yang diinginkan. Nilai beban yang digunakan adalah yang memenuhi batasan *ramp-rate* yang dimiliki setiap unit pembangkitan. Nilai beban juga tidak boleh melebihi kampasitas *maximum* pembangitan dan tidak boleh kurang dari kapasitas *minimum* dari pembangkitan. Form pengisian beban pada setiap periodenya dapat dilihat pada Gambar 3.6 berikut,




**Gambar 3.6** Tampilan pengisian beban

Pada Gambar 3.6 diberikan contoh ketika periode beban yang ingin dilakukan perhitungan DED sebanyak dua periode. Sehingga akan muncul kolom pengisian beban 'Periode 1' dan 'Periode 2'. Tombol 'Ok' ditekan setelah semua kolom pengisian beban disetiap periode telah disikan dan akan mucul form hasil keluaran perhitungan DED. Pada aplikasi ini diberikan pilihan pada 'Consider Ramp-rate', jika menetan pilihan 'Yes' maka DED akan diperhitungkan dengan melihat batasan *ramp-rate* yang telah diisi pada Gambar 3.4. Jika pilihan 'No' ditekan maka perhitungan DED akan dilakukan tanpa memperhatikan batasan *ramp-rate* yang tentu saja sangat memungkinkan hasil perhitungan nantinya akan melanggar batasan *ramp-rate*. Hasil perhitungan akan ditampilkan seperti yang ditunnukkan pada Gambar 3.7.




**Gambar 3.7** Tampilan hasil perhitungan DED

Pada Gambar 3.7 digunakan contoh perhitungan DED 3 unit generator pada 2 periode beban. Tombol 'Save Report' berfungsi untuk menyimpan hasil perhitungan DED yang telah dilakukan. Data yang disimpan akan berupa *file.txt*.

# BAB 4
# ANALISA DAN SIMULASI DATA

Pada Bab ini akan menampilkan hasil simulasi perhitungan *Dynamic Economic Dispatch* dengan menggunkan iterasi lambda. Analisa yang dilakukan adalah melihat pengaruh *ramp-rate* dalam perhitungan DED. Diberikan beberap contoh kasus pada Bab 4 ini, yaitu kasus 1, kasus 2 sebagai uji validasi. Sedangkan kasus 3, kasus 4, kasus 5, dan kasus 6 digunakan untuk melihat pengaruh perhitungan DED dengan memperhatikan batasan *ramp-rate.* Pada kasus 5 dan kasus 6 akan terlihat pengaruh *ramp-rate* yang mengakibatkan selektivitas pembangkitan setiap unit.

## 4.1 Kasus 1

Pada kasus 1 digunakan data pembangkitan melihat dari reverensi [3] pada contoh kasus 3A, dan data beban dilanjutkan pada interval waktu 3 jam kedepan. Diberikan parameter *up-rate* (UR) dan *down_rate* (DR) sebagai batasan ramp-rate. Kasus ini akan melihat tiga periode beban dengan interval waktu satu jam setiap periodenya. Tujuan dari kasus 1 adalah sebagai uji validasi perhitungan DED telah tidak melebihi batasan yang diberikan ketika memperhatikan batasan *ram-rate.* Berikut adalah data yang digunakan dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

**Tabel 4.1** Data karakteristik pembangkit 3 unit

| | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| A | 0.00142 | 0.00194 | 0.0048 |
| B | 7.2 | 7.85 | 7.97 |
| C | 510 | 310 | 78 |
| Pmin | 150 | 100 | 50 |
| Pmax | 600 | 400 | 200 |
| UR | 10 | 30 | 20 |
| DR | 10 | 30 | 20 |
| Fuelcost | 1.1 | 1 | 1 |

**Tabel 4.2** Data beban pembangkit 3 unit

| Jam | 1 | 2 | 3 |
|---|---|---|---|
| Beban | 850 | 800 | 825 |

Setelah dilakukan pengolahan data dengan menggunakan aplikasi perhitungan Delphi yang telah dibuat, maka didapatkan hasil pembangkitan setiap unit pada Tabel 4.3 sebagai berikut:

**Tabel 4.3** Pembangkitan kasus 1 dengan *ramp-rate*

| Jam | Daya Pembangkit (MW) | | |
|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 |
| 1 | 393.2 | 334.6 | 122.2 |
| 2 | 383.2 | 306.1 | 110.7 |
| 3 | 381.4 | 325.1 | 118.4 |

Dari hasil pembangkitan dapat dilihat perubahan setiap unit untuk mencapai beban pada periode berikutnya. Perubahan dapat dilihat pada Tabel 4.4 berikut,

**Tabel 4.4** Perubahan pembangkit/jam kasus 1 dengan *ramp-rate*

| Unit | Jam | |
|---|---|---|
| | 1 ke 2 | 2 ke 3 |
| Unit 1 | -10 | -1.8 |
| Unit 2 | -28.5 | 19 |
| Unit 3 | -11.5 | 7.7 |

Tabel 4.4 menyatakan bahwa perubahan setiap unit pada interval setiap satu jam tidak melebihi batasan *ramp-rate*. Ini berarti aplikasi perhitungan Delphi dapat dijalankan sesuai dengan batasan yang telah diberikan.

Analisa pada kasus 1 dilanjutkan dengan menjalankan DED dengan tidak memperhatikan batasan *ramp-rate*. Didapatkan hasil pembangkitan setiap unit yang berbeda dengan DED yang memperhatikan batasan *ramp-rate*. Hal ini bertujuan untuk melihat hasil keluaran pembangkitan setiap unit pada setiap periode dapat dilakukan DED tanpa melihat

batasan *ramp-rate* akan memiliki pembangkitan yang sama dengan hasil yang ditunjukkan pad Tabel 4.4. Hasil pembangkitan dapat dilihat pada Tabel 4.5 berikut,

**Tabel 4.5** Pembangkitan kasus 1 tanpa *ramp-rate*

| Jam | Daya Pembangkit (MW) | | |
|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 |
| 1 | 393.2 | 334.6 | 122.2 |
| 2 | 369.7 | 315.7 | 114.6 |
| 3 | 381.4 | 325.2 | 118.4 |

Akan tetapi perubahan setiap unit untuk beban setiap jam tidak memenuhi batasan *ramp-rate*, meskipun daya yang dibangkitkan memenuhi batasan *maximum* dan *minimum* pembangkitan setiap unit. Perubahan pembangkitan dapat dilihat pada Tabel 4.6 berikut,

**Tabel 4.6** Perubahan pembangkit/jam kasus 1 tanpa *ramp-rate*

| Unit | Jam | |
|---|---|---|
| | 1 ke 2 | 2 ke 3 |
| Unit 1 | -23.5 | 11.7 |
| Unit 2 | -18.9 | 9.5 |
| Unit 3 | -7.6 | 3.8 |

## 4.2 Kasus 2

Kasus 1 kemudian dilanjutkan dengan melihat *Bloss matrix* yang telah diberikan dan diperhitungkan dengan melihat batasan *ramp-rate*. Pada kasus 2 ini bertujuann untuk melihat hasil perhitungan DED dengan memperhatikan *ramp-rate* telah benar jika diperhitungkan dengan rugi-rugi yang didapatkan dari *Bloss matrix*, dimana nilai pembangkitan diharuskan lebih besar daripada tidak memperhitungkan rugi-rugi.

**Tabel 4.7** *Bloss matrix* kasus 1

| Bij | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0.00003 | 0 | 0 |
| 2 | 0 | 0.00009 | 0 |
| 3 | 0 | 0 | 0.00012 |

Dari penjalana perhitungan dengan aplikasi perhitungan DED yang telah dibuat didapatkan hasil yang dapat dilihat dalam Tabel 4.8 berikut,

**Tabel 4.8** Pembangkitan kasus 1 dengan rugi-rugi

| Jam | Daya Pembangkit (MW) | | |
|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 |
| 1 | 435.2 | 300 | 130.7 |
| 2 | 425.2 | 271.5 | 117 |
| 3 | 421.9 | 291.4 | 126.6 |

Didapatkan hasil pembangkitan setiap unit yang lebih besar daripada pembangkitan unit yang tidak memperhitungkan nilai rugi-rugi pada Tabel 4.3, hal ini dikarenakan rugi-rugi daya menjadi beban tambahan pada penjadwalan pembangkitan.

## 4.3 Kasus 3

Pada kasus 3 data yang digunakan adalah 5 unit pembangkit [8]. DED diperhitngkan dengan data beban yang berubah selama 24 jam. Berikut adalah data karakteristik pembangkit,

**Tabel 4.9** Data karakteristik pembangkit 5 unit

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|---|
| A | 0.008 | 0.003 | 0.0012 | 0.001 | 0.0015 |
| B | 2 | 1.8 | 2.1 | 2 | 1.8 |
| C | 2.5 | 60 | 100 | 120 | 40 |
| Pmin | 10 | 20 | 30 | 40 | 50 |
| Pmax | 75 | 125 | 175 | 250 | 300 |

**Tabel 4.9** Data karakteristik pembangkit 5 unit (lanjutan)

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|---|
| DR | 30 | 30 | 40 | 50 | 50 |
| UR | 30 | 30 | 40 | 50 | 50 |

**Tabel 4.10** Data beban pembangkit 5 unit

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 410 | 9 | 690 | 17 | 558 |
| 2 | 435 | 10 | 704 | 18 | 608 |
| 3 | 475 | 11 | 720 | 19 | 654 |
| 4 | 530 | 12 | 740 | 20 | 704 |
| 5 | 558 | 13 | 704 | 21 | 680 |
| 6 | 608 | 14 | 690 | 22 | 605 |
| 7 | 626 | 15 | 654 | 23 | 527 |
| 8 | 654 | 16 | 580 | 24 | 463 |

Perhitungan dilakukan dua kali yaitu, *Dynamic Economic Dispatch* ketika tidak memperhatikan batasan *ramp-rate* dan *Dynamic Economic Dispatch* dengan memperhatikan batasan *ramp-rate*. Hal ini bertujuan untuk melihat hasil perhitungan dengan adanya efek batasan *ramp-rate*

**Tabel 4.11** Pembangkitan kasus 3 tanpa *ramp-rate*

| Jam | Daya Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 1 | 15.3 | 72.6 | 61.6 | 118.5 | 145.6 |
| 2 | 16.4 | 75.4 | 68.8 | 127 | 151.4 |
| 3 | 18.2 | 80 | 80.4 | 140.6 | 160.7 |
| 4 | 20.6 | 86.3 | 96.4 | 159.3 | 173.5 |
| 5 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |
| 6 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 7 | 24.8 | 97.3 | 124.4 | 192 | 195.8 |

**Tabel 4.11** Pembangkitan kasus 3 tanpa *ramp-rate* (lanjutan)

| Jam | Data Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 8 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 9 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 10 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 11 | 29 | 108.1 | 151.9 | 224.2 | 217.8 |
| 12 | 29.9 | 110.4 | 157.7 | 231 | 222.5 |
| 13 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 14 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 15 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 16 | 22.8 | 92 | 110.9 | 176.3 | 185.1 |
| 17 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |
| 18 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 19 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 20 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 21 | 27.2 | 103.5 | 140.1 | 210.5 | 208.5 |
| 22 | 23.9 | 94.9 | 118.2 | 184.9 | 190.9 |
| 23 | 20.5 | 85.9 | 95.5 | 158.3 | 172.8 |
| 24 | 17.6 | 78.6 | 76.9 | 136.5 | 157.9 |

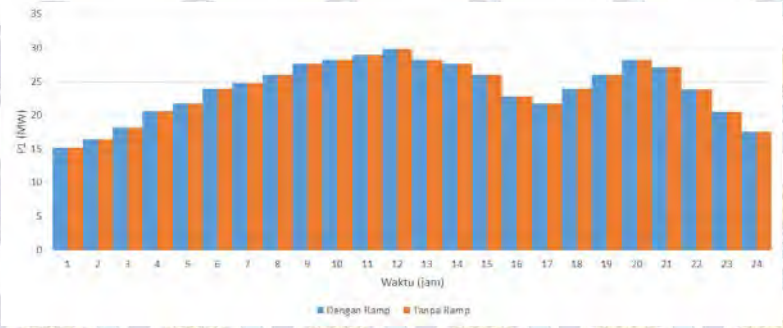
**Tabel 4.12** Pembangkitan kasus 3 dengan *ramp-rate*

| Jam | Daya Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 1 | 15.3 | 72.6 | 61.6 | 118.5 | 145.6 |
| 2 | 16.4 | 75.4 | 68.8 | 127 | 151.4 |
| 3 | 18.2 | 80 | 80.4 | 140.6 | 160.7 |
| 4 | 20.6 | 86.3 | 96.4 | 159.3 | 173.5 |
| 5 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |

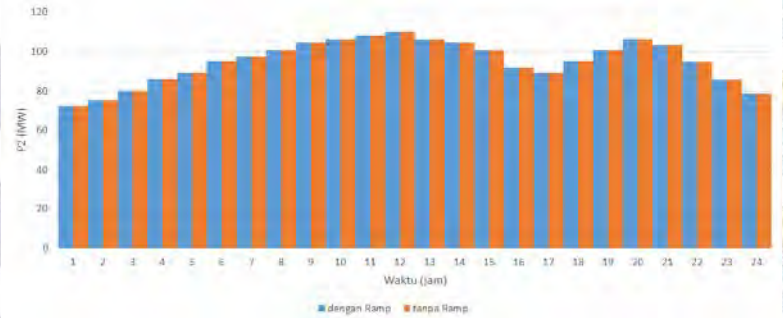
**Tabel 4.12** Pembangkitan kasus 3 dengan *ramp-rate* (lanjutan)

| Jam | Data Pembangkit (MW) | | | | |
|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
| 6 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 7 | 24.8 | 97.3 | 124.4 | 192 | 195.8 |
| 8 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 9 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 10 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 11 | 29 | 108.1 | 151.9 | 224.2 | 217.8 |
| 12 | 29.9 | 110.4 | 157.7 | 231 | 222.5 |
| 13 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 14 | 27.7 | 104.6 | 143.1 | 213.9 | 210.8 |
| 15 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 16 | 22.8 | 92 | 110.9 | 176.3 | 185.1 |
| 17 | 21.8 | 89.5 | 104.5 | 168.8 | 180 |
| 18 | 24 | 95.2 | 119.1 | 185.9 | 191.6 |
| 19 | 26.1 | 100.5 | 132.5 | 201.6 | 202.4 |
| 20 | 28.3 | 106.2 | 147.2 | 218.7 | 214.1 |
| 21 | 27.2 | 103.5 | 140.1 | 210.5 | 208.5 |
| 22 | 23.9 | 94.9 | 118.2 | 184.9 | 190.9 |
| 23 | 20.5 | 85.9 | 95.5 | 158.3 | 172.8 |
| 24 | 17.6 | 78.6 | 76.9 | 136.5 | 157.9 |

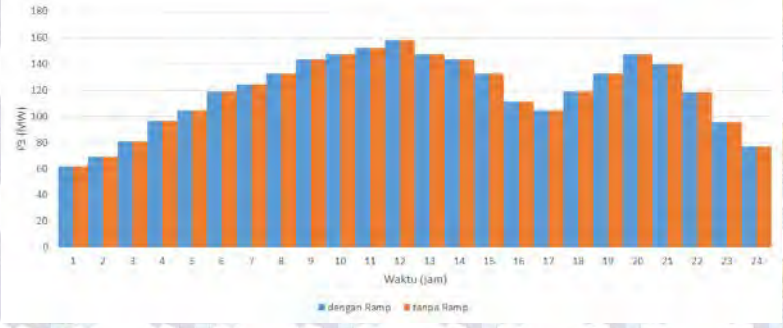
Dari data yang didapatkan dari Tabel 4.11 dan Tabel 4.12 dibuat garafik perubahan beban setiap unit. Grafik setiap unit dilihat perubahan kenaikan dan kenaikan pada setiap jamnya. Perubahan dapat dilihat pada Gambar 4.1 hingga Gambar 4.5.



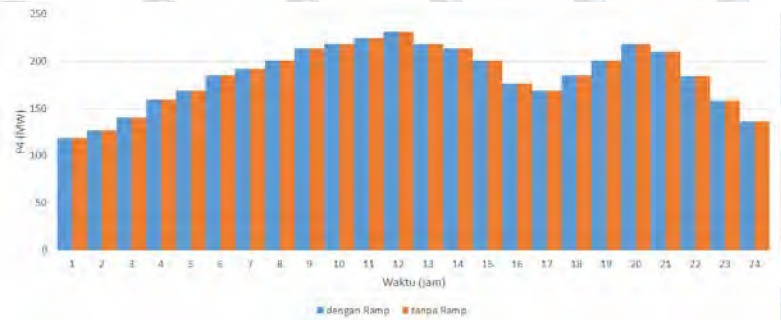

**Gambar 4.1** Grafik pembangkitan unit 1 kasus 3



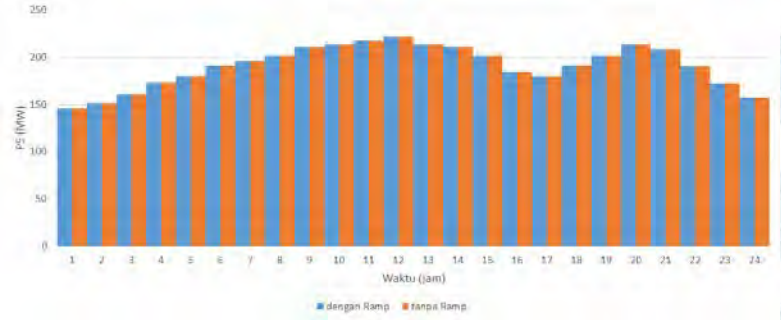

**Gambar 4.2** Grafik pembangkitan unit 2 kasus 3




**Gambar 4.3** Grafik pembangkitan unit 3 kasus 3




**Gambar 4.4** Grafik pembangkitan unit 4 kasus 3




**Gambar 4.5** Grafik pembangkitan unit 5 kasus 3

**Tabel 4.13** Perbandingan total biaya kasus 3

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 40121.08 |
| Dengan *Ramp-rate* | 40121.08 |

Dari hasil data pembangkitan yang didapatkan, terlihat tidak adanya perubahan pembangkitan setiap unit ketika tidak memperhatikan *ramp-rate* dan dengan memperhatikan *ramp-rate*. Hal ini dikarenakan nilai optimal dari unit yang memperhatikan *ramp-rate* masih berada didalam batasan pembangkitan *maximum* dan *minimum* generator, sehingga tidak terjadi seleksi penjadwalan pembangkit dan total biaya yang dihasilkan sama. Hal ini dapat dilihat pada Tabel 4.13.

## 4.4 Kasus 4

Kasus 4 menggunakan data karakteristik 6 pembangkit [9]. Sama dengan kasus 3, pada kasus 4 akan dilakukan perhitunga sebanyak dua kali yaitu ketika tidak memperhatikan batasan *ramp-rate*, dan perhitungan ketika memperhatikan batasan *ramp-rate*. Data yang digunakan adalah sebagai berikut,

**Tabel 4.14** Data karakteristik pembangkit 6 unit

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 |
|---|---|---|---|---|---|---|
| A | 0.007 | 0.0095 | 0.009 | 0.009 | 0.008 | 0.0075 |
| B | 7 | 10 | 8 | 11 | 10.5 | 12 |
| C | 240 | 200 | 220 | 200 | 220 | 190 |
| Pmin | 100 | 50 | 80 | 50 | 50 | 50 |
| Pmax | 500 | 200 | 300 | 150 | 200 | 120 |
| DR | 120 | 90 | 100 | 90 | 90 | 90 |
| UR | 80 | 50 | 65 | 50 | 50 | 50 |

**Tabel 4.15** Data beban pembangkitan 6 unit

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 955 | 9 | 1126 | 17 | 1221 |
| 2 | 942 | 10 | 1150 | 18 | 1202 |
| 3 | 953 | 11 | 1201 | 19 | 1159 |
| 4 | 930 | 12 | 1235 | 20 | 1092 |
| 5 | 935 | 13 | 1190 | 21 | 1023 |
| 6 | 963 | 14 | 1251 | 22 | 984 |
| 7 | 989 | 15 | 1263 | 23 | 975 |
| 8 | 1023 | 16 | 1250 | 24 | 960 |

Dari hasil perhitungan data Tabel 4.14 dan data Tabel 4.15 didapatkan hasil pembangkitan sebagai berikut.

**Tabel 4.16** Pembangkitan kasus 4 tanpa *ramp-rate*

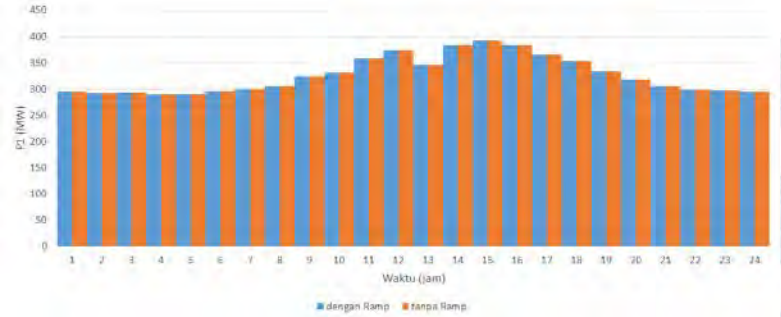
| Jam | Daya Pembangkit (MW) | | | | | |
|---|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 |
| 1 | 294.2 | 95.9 | 260.5 | 150 | 149.8 | 51.4 |
| 2 | 291.6 | 93.6 | 257 | 150 | 145.7 | 50 |
| 3 | 293.9 | 95.6 | 260 | 150 | 149.2 | 50.9 |
| 4 | 288.8 | 91.1 | 253.5 | 150 | 141.6 | 50 |
| 5 | 290 | 92.1 | 255 | 150 | 143.3 | 50 |
| 6 | 295.6 | 97.2 | 262.5 | 150 | 151.9 | 53.2 |
| 7 | 300 | 101.2 | 268.9 | 150 | 159 | 59.1 |
| 8 | 305.8 | 106.5 | 277.3 | 150 | 168.2 | 66.9 |
| 9 | 324 | 123.2 | 300 | 150 | 197.5 | 91.2 |
| 10 | 331.7 | 130.3 | 300 | 150 | 200 | 100.9 |
| 11 | 358.1 | 154.6 | 300 | 150 | 200 | 120 |
| 12 | 374 | 169.2 | 300 | 150 | 200 | 120 |
| 13 | 346 | 143.6 | 300 | 150 | 200 | 119 |
| 14 | 384.3 | 178.7 | 300 | 150 | 200 | 120 |
| 15 | 392.1 | 185.9 | 300 | 150 | 200 | 120 |
| 16 | 383.7 | 178.1 | 300 | 150 | 200 | 120 |
| 17 | 365.1 | 161 | 300 | 150 | 200 | 120 |
| 18 | 353 | 149.9 | 300 | 150 | 200 | 120 |
| 19 | 334 | 133.2 | 300 | 150 | 200 | 104.9 |
| 20 | 317.5 | 117.2 | 294.4 | 150 | 187.2 | 82.7 |
| 21 | 305.8 | 106.5 | 277.3 | 150 | 168.3 | 66.9 |
| 22 | 299.2 | 100.4 | 267.6 | 150 | 157.6 | 58 |
| 23 | 297.7 | 99 | 265.4 | 150 | 155.2 | 55.9 |
| 24 | 295.1 | 96.7 | 261.8 | 150 | 151.1 | 52.5 |

**Tabel 4.17** Pembangkitan kasus 4 dengan *ramp-rate*

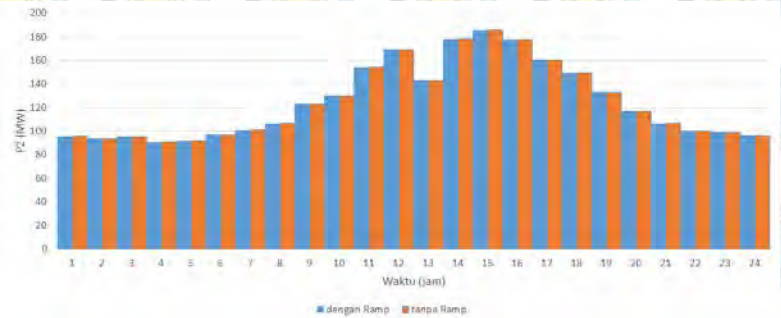
| Jam | Daya Pembangkit (MW) | | | | | |
|---|---|---|---|---|---|---|
| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 | Unit 6 |
| 1 | 294.2 | 95.9 | 260.5 | 150 | 149.8 | 51.4 |
| 2 | 291.6 | 93.6 | 257 | 150 | 145.7 | 50 |
| 3 | 293.9 | 95.6 | 260 | 150 | 149.2 | 50.9 |
| 4 | 288.8 | 91.1 | 253.5 | 150 | 141.6 | 50 |
| 5 | 290 | 92.1 | 255 | 150 | 143.3 | 50 |
| 6 | 295.6 | 97.2 | 262.5 | 150 | 151.9 | 53.2 |
| 7 | 300 | 101.2 | 268.9 | 150 | 159 | 59.1 |
| 8 | 305.8 | 106.5 | 277.3 | 150 | 168.2 | 66.9 |
| 9 | 324 | 123.2 | 300 | 150 | 197.5 | 91.2 |
| 10 | 331.7 | 130.3 | 300 | 150 | 200 | 100.9 |
| 11 | 358.1 | 154.6 | 300 | 150 | 200 | 120 |
| 12 | 374 | 169.2 | 300 | 150 | 200 | 120 |
| 13 | 346 | 143.6 | 300 | 150 | 200 | 119 |
| 14 | 384.3 | 178.7 | 300 | 150 | 200 | 120 |
| 15 | 392.1 | 185.9 | 300 | 150 | 200 | 120 |
| 16 | 383.7 | 178.1 | 300 | 150 | 200 | 120 |
| 17 | 365.1 | 161 | 300 | 150 | 200 | 120 |
| 18 | 353 | 149.9 | 300 | 150 | 200 | 120 |
| 19 | 334 | 133.2 | 300 | 150 | 200 | 104.9 |
| 20 | 317.5 | 117.2 | 294.4 | 150 | 187.2 | 82.7 |
| 21 | 305.8 | 106.5 | 277.3 | 150 | 168.3 | 66.9 |
| 22 | 299.2 | 100.4 | 267.6 | 150 | 157.6 | 58 |
| 23 | 297.7 | 99 | 265.4 | 150 | 155.2 | 55.9 |
| 24 | 295.1 | 96.7 | 261.8 | 150 | 151.1 | 52.5 |

Didapatkan dari Tabel 4.16 dan Tabel 4.17 dibuat garafik perubahan beban setiap unit. Grafik setiap unit dilihat perubahan kenaikan dan penurunan pembangkitan pada setiap jamnya.

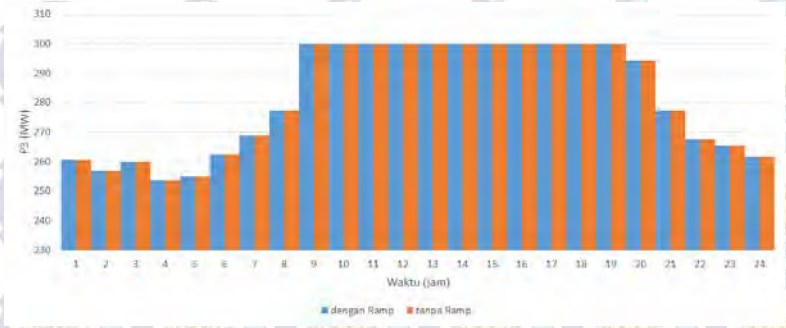
Pada gambar grafik didapatkan bentuk pola naik dan turunnya pembangkitan adalah sama ketika tidak memperhatikan *ramp-rate* dan ketika memperhatikan *ramp-rate*. Pola ini sama dengan contoh kasus 3. Untuk lebih jelasnya dapat dilihat pada Gambar 4.6 sampai 4.11 berikut.



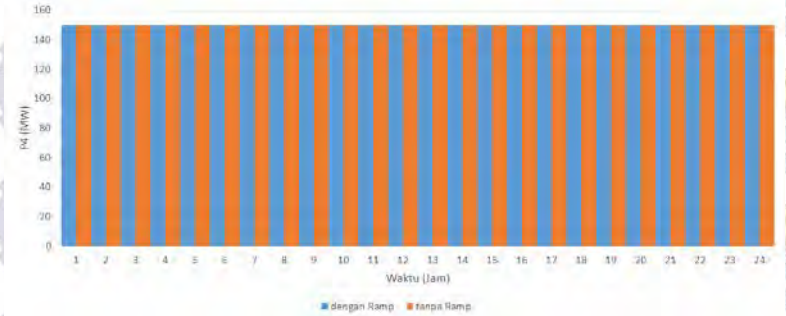

**Gambar 4.6** Grafik pembangkitan unit 1 kasus 4



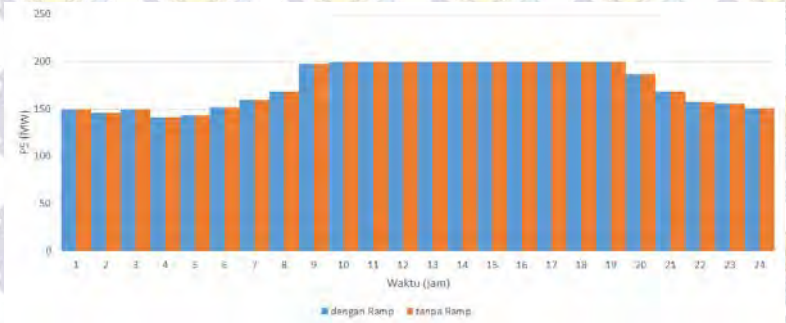

**Gambar 4.7** Grafik pembangkitan unit 2 kasus 4



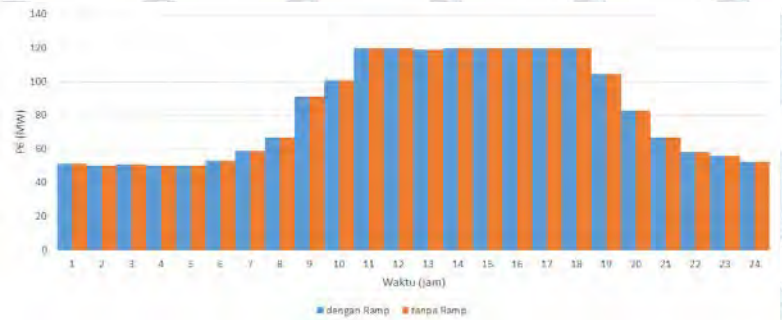

**Gambar 4.8** Grafik pembangkitan unit 3 kasus 4




**Gambar 4.9** Grafik pembangkitan unit 4 kasus 4




**Gambar 4.10** Grafik pembangkitan unit 5 kasus 4




**Gambar 4.11** Grafik pembangkitan unit 6 kasus 4

**Tabel 4.18** Perbandingan total biaya kasus 4

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 328724.3 |
| Dengan *Ramp-rate* | 328724.3 |

Dari data yang didapatkan menghasilkan pembangikitan yang sama ketika tidak memperhatikan *ramp-rate* dan ketika memperhatikan *ramp-rate*. Hal ini dikarenakan nilai optimal dari unit yang memperhatikan *ramp-rate* masih berada didalam batasan pembangkitan *maximum* dan *minimum* generator, sehingga tidak terjadi seleksi perubahan penjadwalan pembangkit dan total biaya pembangkitan yang dihasilkan sama. Dapat dilihat pada Tabel 4.18

## 4.5 Kasus 5

Digunakan data 10 unit [10] pembangkit. *Dynamic Economic Dispatch* diperhitung terhadap perubahan beban selama 24 jam. Pada kasus 5 juga melakukan dua kali perhitungan, yaitu DED tanpa memperhatikan *ramp-rate* dan DED dengan memperhatikan *ramp-rate*. Hal ini bertujuan untuk melihat pengaruh ramp-rate dalam perhitungan DED. Data karakteristik pembangkit yang digunakan adalah sebagai berikut,

**Tabel 4.19** Data karakteristik pembangkit 10 Unit

| | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|---|
| A | 0.1524 | 0.1058 | 0.028 | 0.0354 | 0.0211 |
| B | 38.5397 | 46.1591 | 40.3965 | 38.3055 | 36.3278 |
| C | 786.7988 | 451.3251 | 1049.9977 | 1243.5311 | 1658.5696 |
| Pmin | 150 | 135 | 73 | 60 | 73 |
| Pmax | 470 | 470 | 340 | 300 | 243 |
| DR | 80 | 80 | 80 | 50 | 50 |
| UR | 80 | 80 | 80 | 50 | 50 |
| | Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| A | 0.0179 | 0.0121 | 0.0121 | 0.109 | 0.1295 |
| B | 38.2704 | 36.5104 | 36.5104 | 39.5804 | 40.5407 |
| C | 1356.6592 | 1450.7045 | 1450.7045 | 1455.6056 | 1469.4026 |
| Pmin | 57 | 20 | 47 | 20 | 10 |
| Pmax | 160 | 130 | 120 | 80 | 55 |
| DR | 50 | 30 | 30 | 30 | 30 |
| UR | 50 | 30 | 30 | 30 | 30 |

**Tabel 4.20** Data beban pembangkitan 10 unit kasus 5

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 1036 | 9 | 1924 | 17 | 1480 |
| 2 | 1110 | 10 | 2022 | 18 | 1628 |
| 3 | 1258 | 11 | 2106 | 19 | 1776 |
| 4 | 1406 | 12 | 2150 | 20 | 1936 |
| 5 | 1480 | 13 | 2072 | 21 | 1924 |
| 6 | 1628 | 14 | 1924 | 22 | 1628 |
| 7 | 1702 | 15 | 1776 | 23 | 1332 |
| 8 | 1776 | 16 | 1554 | 24 | 1184 |

Dari perhitungan data Tabel 4.19 dan data Tabel 4.20 didapatkan

**Tabel 4.21** Pembangkitan kasus 5 tanpa *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |
| 3 | 73 | 79.6 | 126.9 | 192.3 | 224.9 | 293.8 |
| 4 | 74.8 | 92.5 | 129.9 | 181.6 | 207.5 | 261.9 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 20 | 24.2 | 36.3 | 53.1 | 61.5 | 79.2 |
| 10 | 11.8 | 16.7 | 26.9 | 41 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 218 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.6 | 278 | 335.9 | 385.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.6 | 218 | 157.4 | 150 | 150 | 150 |
| 2 | 365.4 | 278 | 190.6 | 135 | 135 | 135 |

**Tabel 4.21** Pembangkitan kasus 5 tanpa *ramp-rate* (lanjutan)

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 3 | 340 | 340 | 340 | 257.7 | 224.9 | 293.8 |
| 4 | 300 | 300 | 300 | 233.4 | 207.5 | 261.9 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 69.9 | 61.5 | 79.2 |
| 10 | 55 | 55 | 55 | 55 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 157.4 | 222.9 | 218 | 150 | 150 | 150 |
| 2 | 190.6 | 285.1 | 278 | 135 | 135 | 135 |
| 3 | 340 | 340 | 340 | 293.8 | 159.6 | 100.2 |
| 4 | 300 | 300 | 300 | 261.9 | 155.7 | 108.8 |
| 5 | 243 | 243 | 243 | 243 | 243 | 229.4 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 79.2 | 44.7 | 29.5 |
| 10 | 55 | 55 | 55 | 55 | 33.9 | 21.1 |

**Tabel 4.22** Pembangkitan kasus 5 dengan *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |

**Tabel 4.22** Pembangkitan kasus 5 dengan *ramp-rate* (lanjutan)

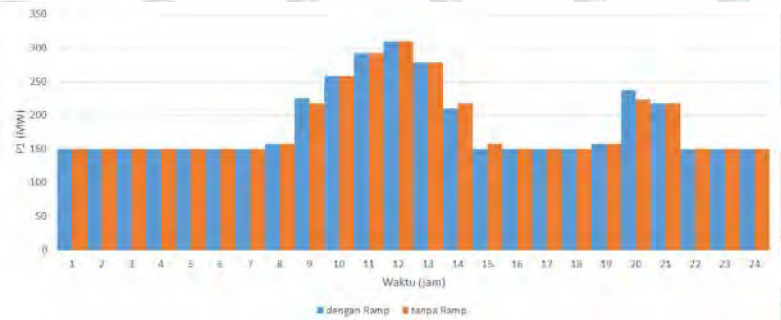
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 73 | 79.6 | 126.9 | 193.4 | 224.9 | 297.5 |
| 4 | 74.8 | 92.5 | 129.9 | 179.9 | 207.5 | 257.5 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 20 | 24.2 | 36.3 | 53.4 | 61.5 | 80 |
| 10 | 11.8 | 16.7 | 26.9 | 41.3 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 225.4 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.7 | 270.7 | 335.9 | 285.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.6 | 210.6 | 150 | 150 | 150 | 150 |
| 2 | 365.4 | 285.4 | 205.4 | 135 | 135 | 135 |
| 3 | 340 | 340 | 336.7 | 256.7 | 224.9 | 297.5 |
| 4 | 300 | 300 | 295.9 | 245.9 | 207.5 | 257.5 |

**Tabel 4.22** Pembangkitan kasus 5 dengan *ramp-rate* (lanjutan)

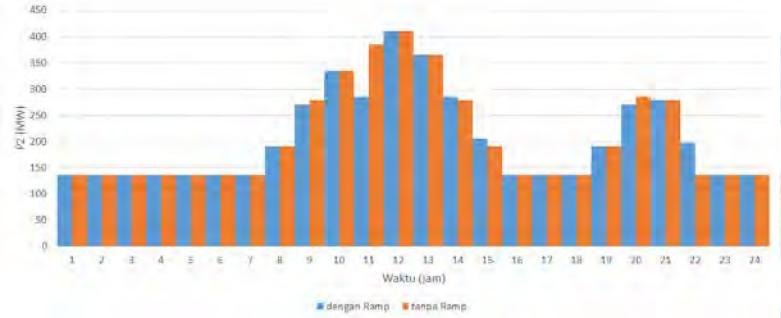
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 63.6 | 61.5 | 80 |
| 10 | 55 | 55 | 55 | 49.8 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 157.4 | 237.4 | 218 | 150 | 150 | 150 |
| 2 | 190.7 | 270.7 | 278 | 198 | 135 | 135 |
| 3 | 340 | 340 | 340 | 260 | 180 | 100 |
| 4 | 300 | 300 | 300 | 250 | 200 | 150 |
| 5 | 243 | 243 | 243 | 243 | 200 | 150 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 65.5 | 35.5 | 23.6 |
| 10 | 55 | 55 | 55 | 51.5 | 21.5 | 16.2 |

Didapatkan dari Tabel 4.21 dan Tabel 4.22 dibuat garafik perubahan beban setiap unit. Grafik setiap unit dilihat perubahan kenaikan dan penurunan pada setiap jamnya.

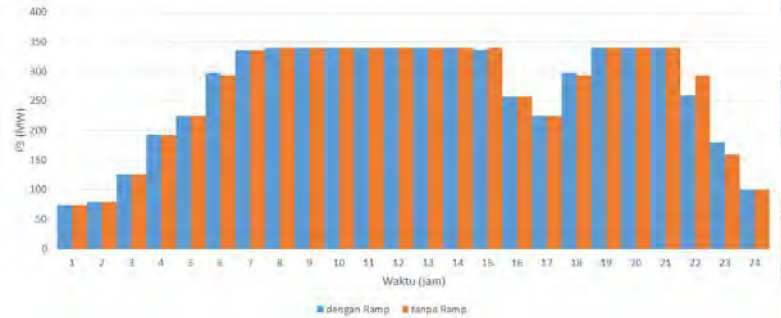
Pada gambar grafik didapatkan bentuk pola ketika tidak memperhatikan *ramp-rate* dapat naik dan turun dengan dengan melanggar batasan *ramp-rate*, sedangkan ketika memperhatikan *ramp-rate* pola naik dan turun pembangkit bertahap sesuai dengan nilai DR dan UR. Untuk lebih jelasnya dapat dilihat pada Gambar 4.12 sampai 4.16 berikut.



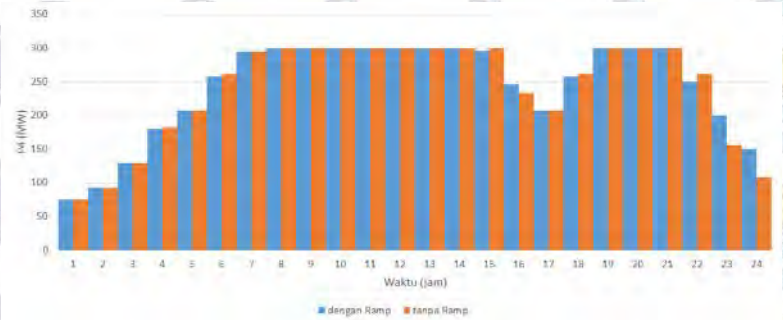

**Gambar 4.12** Grafik pembangkitan unit 1 kasus 5



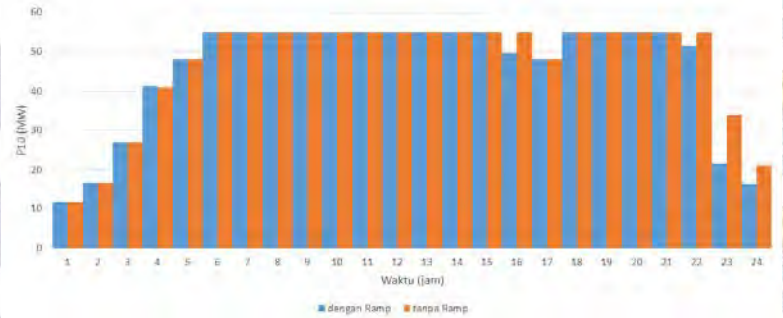

**Gambar 4.13** Grafik pembangkitan unit 2 kasus 5




**Gambar 4.14** Grafik pembangkitan unit 3 kasus 5




**Gambar 4.15** Grafik pembangkitan unit 4 kasus 5




**Gambar 4.16** Grafik pembangkitan unit 10 kasus 5

**Tabel 4.23** Perbandingan total biaya kasus 5

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 2298227 |
| Dengan *Ramp-rate* | 2300521 |

Dari hasil data yang didapatkan, terdapat perbedaan pembangkitan ketika tidak memperhitungkan *ramp-rate* dengan pembangkitan ketika memperhatikan *ramp-rate*. Hal ini dikarenakan batasan pemabangkitan yang melihat pengaruh *ramp-rate* akan semakin dipersempit dan seleksi penjadwalan pembangkitan akan semakin ketat.

Dengan adanya pengaruh ramp-rate pada kasus 5 ini didapatkan hasil total biaya yang lebih mahal $2294 daripada total biaya ketika tidak memperhatikan batasan *ramp-rate*.

## 4.6 Kasus 6

Pada contoh kasus 6 ini merupakan lanjutan dari kasus 5. Data karakteristik pembangkit yang digunakan sama dengan data yang ditampilkan pada Tabel 4.17. Sedangkan data beban yang digunakan pada kasus 6 dilakukan perubahan pada jam ke 19 dapat dilihat pada Tabel 4.24, dimana nantinya akan melihat efek perubahan penjadwalan pembangkit pada suatu periode dikarenakan *ramp-rate* nilai kombinasi *minimum* pembangkitan harus dilakukan perubahan untuk bisa memenuhi beban untuk interval waktu berikutnya.

**Tabel 4.24** Data beban pembangkit 10 unit kasus 6

| Jam | Beban | Jam | Beban | Jam | Beban |
|---|---|---|---|---|---|
| 1 | 1036 | 9 | 1924 | 17 | 1480 |
| 2 | 1110 | 10 | 2022 | 18 | 1628 |
| 3 | 1258 | 11 | 2106 | 19 | 1776 |
| 4 | 1406 | 12 | 2150 | 20 | 1972 |
| 5 | 1480 | 13 | 2072 | 21 | 1924 |
| 6 | 1628 | 14 | 1924 | 22 | 1628 |
| 7 | 1702 | 15 | 1776 | 23 | 1332 |
| 8 | 1776 | 16 | 1554 | 24 | 1184 |

Dari perhitungan data Tabel 4.19 dan data Tabel 4.24 didapatkan

**Tabel 4.25** Pembangkitan kasus 6 tanpa *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |
| 3 | 73 | 79.6 | 126.9 | 192.3 | 224.9 | 293.8 |
| 4 | 74.8 | 92.5 | 129.9 | 181.6 | 207.5 | 261.9 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |

**Tabel 4.25** Pembangkitan kasus 6 tanpa *ramp-rate* (lanjutan)

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 20 | 24.2 | 36.3 | 53.1 | 61.5 | 79.2 |
| 10 | 11.8 | 16.7 | 26.9 | 41 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 218 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.6 | 278 | 335.9 | 385.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.6 | 218 | 157.4 | 150 | 150 | 150 |
| 2 | 365.4 | 278 | 190.6 | 135 | 135 | 135 |
| 3 | 340 | 340 | 340 | 257.7 | 224.9 | 293.8 |
| 4 | 300 | 300 | 300 | 233.4 | 207.5 | 261.9 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |

**Tabel 4.25** Pembangkitan kasus 6 tanpa *ramp-rate* (lanjutan)

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 9 | 80 | 80 | 80 | 69.9 | 61.5 | 79.2 |
| 10 | 55 | 55 | 55 | 55 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 157.4 | 237.4 | 218 | 150 | 150 | 150 |
| 2 | 190.6 | 306.3 | 278 | 135 | 135 | 135 |
| 3 | 340 | 340 | 340 | 293.8 | 159.6 | 100.2 |
| 4 | 300 | 300 | 300 | 261.9 | 155.7 | 108.8 |
| 5 | 243 | 243 | 243 | 243 | 243 | 229.4 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 79.2 | 44.7 | 29.5 |
| 10 | 55 | 55 | 55 | 55 | 33.9 | 21.1 |

**Tabel 4.26** Pembangkitan kasus 6 dengan *ramp-rate*

| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 150 | 150 | 150 | 150 | 150 | 150 |
| 2 | 135 | 135 | 135 | 135 | 135 | 135 |
| 3 | 73 | 79.6 | 126.9 | 193.4 | 224.9 | 297.5 |
| 4 | 74.8 | 92.5 | 129.9 | 179.9 | 207.5 | 257.5 |
| 5 | 172.4 | 202.1 | 243 | 243 | 243 | 243 |
| 6 | 149 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |

**Tabel 4.26** Pembangkitan kasus 6 dengan *ramp-rate* (lanjutan)

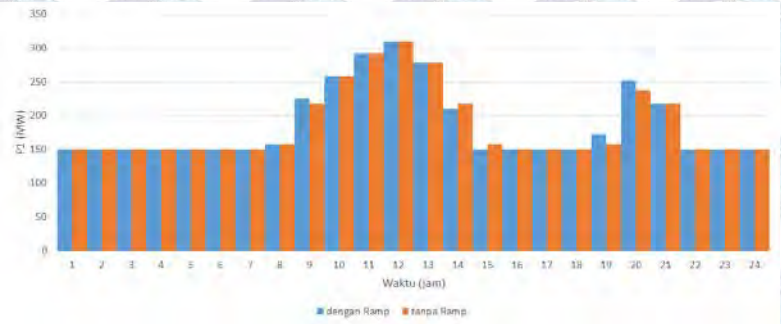
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| 9 | 20 | 24.2 | 36.3 | 53.4 | 61.5 | 80 |
| 10 | 11.8 | 16.7 | 26.9 | 41.3 | 48.1 | 55 |
| Unit | Jam | | | | | |
| | 7 | 8 | 9 | 10 | 11 | 12 |
| 1 | 150 | 157.4 | 225.4 | 258.2 | 292.6 | 310.6 |
| 2 | 135 | 190.7 | 270.7 | 335.9 | 385.4 | 411.4 |
| 3 | 334.7 | 340 | 340 | 340 | 340 | 340 |
| 4 | 294.3 | 300 | 300 | 300 | 300 | 300 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 80 | 80 | 80 |
| 10 | 55 | 55 | 55 | 55 | 55 | 55 |
| Unit | Jam | | | | | |
| | 13 | 14 | 15 | 16 | 17 | 18 |
| 1 | 278.16 | 210.6 | 150 | 150 | 150 | 150 |
| 2 | 365.4 | 285.4 | 205.4 | 135 | 135 | 135 |
| 3 | 340 | 340 | 336.7 | 256.7 | 224.9 | 297.5 |
| 4 | 300 | 300 | 295.9 | 245.9 | 207.5 | 257.5 |
| 5 | 243 | 243 | 243 | 243 | 243 | 243 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 63.6 | 61.5 | 80 |
| 10 | 55 | 55 | 55 | 49.8 | 48.1 | 55 |

**Tabel 4.26** Pembangkitan kasus 6 dengan *ramp-rate* (lanjutan)

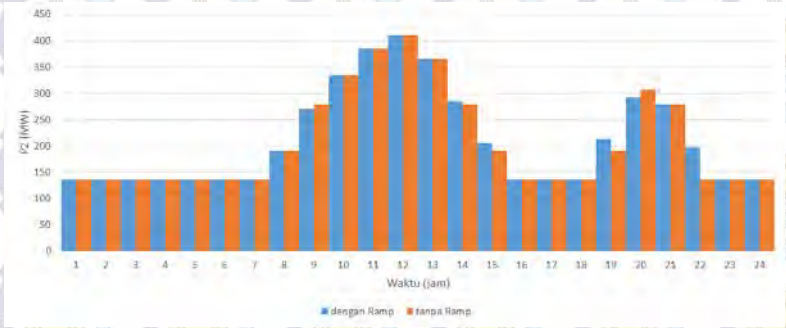
| Unit | Jam | | | | | |
|---|---|---|---|---|---|---|
| | 19 | 20 | 21 | 22 | 23 | 24 |
| 1 | 172.1 | 252.1 | 218 | 150 | 150 | 150 |
| 2 | 211.9 | 291.9 | 278 | 198 | 135 | 135 |
| 3 | 304 | 340 | 340 | 260 | 180 | 100 |
| 4 | 300 | 300 | 300 | 250 | 200 | 150 |
| 5 | 243 | 243 | 243 | 243 | 200 | 199.2 |
| 6 | 160 | 160 | 160 | 160 | 160 | 160 |
| 7 | 130 | 130 | 130 | 130 | 130 | 130 |
| 8 | 120 | 120 | 120 | 120 | 120 | 120 |
| 9 | 80 | 80 | 80 | 65.5 | 35.5 | 23.6 |
| 10 | 55 | 55 | 55 | 51.5 | 21.5 | 16.2 |

Dibuat grafik perubahan kenaikan dan penurunan pembangkitan dari data Tabel 4.25 dan Tabel 4.26.

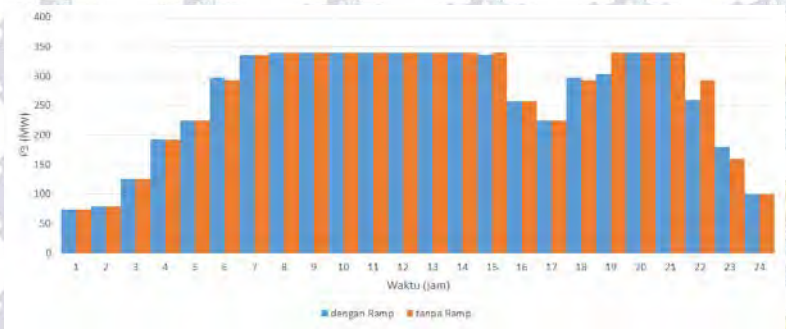
Pada gambar grafik didapatkan bentuk pola ketika tidak memperhatikan *ramp-rate* dapat naik dan turun dengan dengan melanggar batasan *ramp-rate*, sedangkan ketika memperhatikan *ramp-rate* pola naik dan turun pembangkit bertahap sesuai dengan nilai DR dan UR. Untuk lebih jelasnya dapat dilihat pada Gambar 4.17 sampai 4.21 berikut.



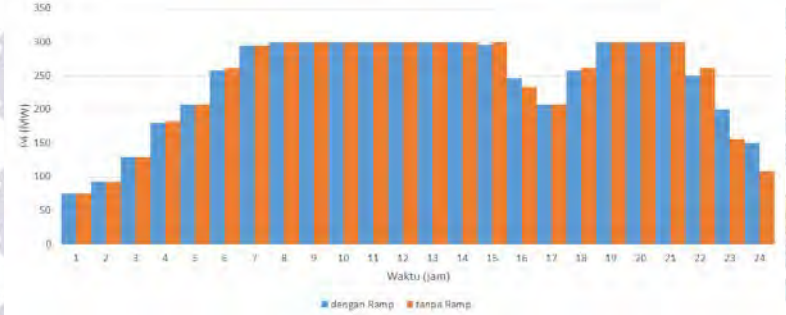

**Gambar 4.17** Grafik pembangkitan unit 1 kasus 6



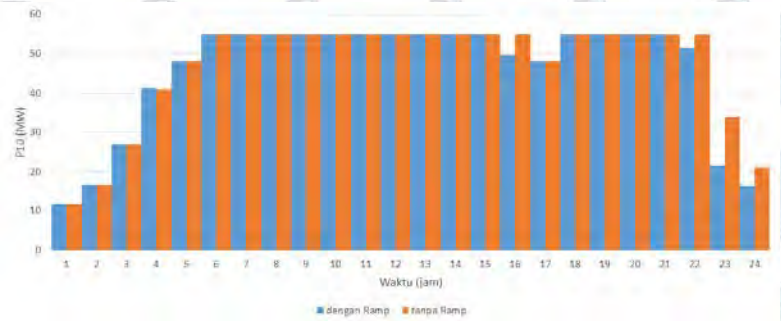

**Gambar 4.18** Grafik pembangkitan unit 2 kasus 6




**Gambar 4.19** Grafik pembangkitan unit 3 kasus 6




**Gambar 4.20** Grafik pembangkitan unit 4 kasus 6




**Gambar 4.21** Grafik pembangkitan unit 10 kasus 6

Pada kasus 6 ini terdapat kondisi dimana pada perhitungan DED beban jam ke 19, nilai kombinasi pembangkitan harus dirubah. Hali ini dikarenakan nilai pembangkitan yang didapatkan untuk jam ke 19 tidak dapat mengejar nilai pembangkitan pada jam ke 20 dikarenakan efek batasan *ramp-rate*. Untuk lebih jelasnya dapat dilihat pada tabel berikut,

**Tabel 4.27** Pembangkitan jam ke 19 sebelum dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 157.4 | 190.7 | 340 | 300 | 243 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 160 | 130 | 120 | 80 | 55 |

Sehingga dengan nilai pembangkitan yang didapatkan pada Tabel 4.27, maka akan didapatkan kemampuan *maximum* akibat *ramp-rate* tiap unit untuk naik pada jam ke 20 dapat dilihat pada Tabel 4.28

**Tabel 4.28** Nilai total *maximum* naik sebelum dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 80 | 80 | 0 | 0 | 0 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 0 | 0 | 0 | 0 | 0 |
| Total Kenaikan | | | | |
| 160 | | | | |

Untuk mencapai beban pada jam ke 20 sebesar 1972, dibutuhkkan kenaikan minimal sebesar 196. Jika nilai optimum pembangkitan pada jam ke 6 tidak dirubah maka tidak memungkinkan untuk bisa melanjutkan perhitungan DED pada jam ke 20, dikarenakan total kenaikan hanya sebesar 160. Oleh karena itu pembangkitan jam ke 6 dirubah menjadi Tabel 4.29 berikut.

**Tabel 4.29** Pembangkitan jam ke 19 setelah dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 172.1 | 211.9 | 304 | 300 | 243 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 160 | 130 | 120 | 80 | 55 |

Dari data Tabel 4.30 memungkinkan untuk melakukan perhitungan DED pada jam ke 20, dikarenakan total kenaikan sebesar 196 memenuhi

**Tabel 4.30** Nilai total *maximum* naik setelah dirubah

| Unit 1 | Unit 2 | Unit 3 | Unit 4 | Unit 5 |
|---|---|---|---|---|
| 80 | 80 | 36 | 0 | 0 |
| Unit 6 | Unit 7 | Unit 8 | Unit 9 | Unit 10 |
| 0 | 0 | 0 | 0 | 0 |
| Total Kenaikan | | | | |
| 196 | | | | |

**Tabel 4.31** Perbandingan total biaya kasus 6

| DED | Total Biaya |
|---|---|
| Tanpa *Ramp-rate* | 2302142 |
| Dengan *Ramp-rate* | 2305528 |

Dari semua data yang ditunjukkan pada kasus 6, menunjukkan efek batasan *ramp-rate* yang membuat nilai pembangkitan setiap generator semakin selektif. Hal ini berakibat nilia total biaya yang dikeluarkan akan lebih mahal $3386 dari pada total biaya dengan tidak melihat batasan *ramp-rate*. Hal ini dapat dilihat pada Tabel 4.31.

# BAB 5
# PENUTUP

## 5.1 Kesimpulan

Dari semua proses yang meliputi studi litealatur, serta simulasi dan analisis, maka terdapat beberapa hal yang dapat disimpullkan terkait Tugas Akhir ini, yaitu:

1. Aplikasi perhitungan DED dapat melakukan perhitungan sesuai dengan batasan tanpa melanggar *ramp-rate* yang telah ditentukan dengan menggunakan metode iterasi lambda dalam waktu tertentu, baik dengan memperhitungkan rugi-rugi yang didapatkan dalam formulasi *Bloss matrix* atau tanpa memperhitungkan rugi-rugi.
2. Adanya pengaruh batasan *ramp-rate* menyebabakan hasil perhitungan dalam DED semakin selektif. Dikarenakan nilai pembangkitan tidak bisa naik dan turun melebihi batasan *ramp-rate* yang telah ditetapkan pada setiap unit pembangkitan.
3. Dalam simulasi pengoperasisn aplikasi perhitungan DED, dengan adanya selektifitas penjadwalan optimum pembangkitan akibat batasan *ramp-rate* total biaya pembangkitan akan menjadi lebih mahal jika dibandingkan ketika melakukan perhitungan DED tanpa memperhatikan nilai *ramp-rate*. Hal ini terlihat dalam simulasi kasus 5 Tabel 4.23 memperlihatkan pengaruh batasan ramp-rate membuat total biaya operasi lebih mahal $2294, dan pada kasus 6 Tabel 4.31 dengan memperhatikan batasa *ramp-rate* lebih mahal $3386.
4. Dalam simulasi kasus 6 diperlukan penggantian nilai penjadwalan pembangkit, dikarenakan oleh pengaruh *ramp-rate* yang menyebab ketidakmampuan nilai optimal pembangkitan untuk melakukan pembangkitan optimal untuk periode beban berikutnya. Penggantian nilai pembangkitan dapat dilakukan dengan perhitungan *Economic Dispatch* pada satu interval waktu.
5. Hasil akhir dari aplikasi perhitungan yang dikembangkan dapat digunakan untuk meng-*upgrade* aplikasi perhitungan yang selama ini digunakan pada mata kuliah Operasi Optimum Sistem Tenaga Litrik

## 5.2 Saran

Adapun saran untuk penelitian selanjutnya yang berkaitan dengan Tugas Akhir ini, yaitu:

1. Aplikasi perhitungan DED dapat dikembangkan dengan memberikan perhitungan rugi-rugi dengan menggunakan *Optimal Power Flow (OPF)*, agar hasil perhitungan lebih optimal.
2. Aplikasi DED dapat dikembanggkan dengan memberikan metode perhitungan yang berbeda sehingga dapat dibandingkan metode iterasi lambda.

# LAMPIRAN

**Tabel L.1 *Bloss matrix* Kasus 3**

10*e-4

| Bij | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.49 | 0.14 | 0.15 | 0.15 | 0.2 |
| 2 | 0.14 | 0.45 | 0.16 | 0.2 | 0.18 |
| 3 | 0.15 | 0.16 | 0.39 | 0.1 | 0.12 |
| 4 | 0.15 | 0.2 | 0.1 | 0.4 | 0.14 |
| 5 | 0.2 | 0.18 | 0.12 | 0.14 | 0.35 |

**Tabel L.1 *Bloss matrix* Kasus 4**

| Boo |
|---|
| 0.00154 |

| | Boi | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 10*e-5 | | 0.38 | 1.79 | -5.32 | 1.52 | 2.33 | 1.26 |

| | Bij | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 10*e-4 | 1 | 2.231 | 1.162 | -0.122 | -0.017 | 0.113 | 0.39 |
| | 2 | 1.162 | 1.89 | -0.077 | -0.048 | 0.069 | 0.28 |
| | 3 | -0.122 | -0.077 | 2.004 | -0.74 | -0.724 | -0.599 |
| | 4 | -0.017 | -0.048 | -0.74 | -1.479 | 0.538 | 0.342 |
| | 5 | 0.113 | 0.069 | -0.724 | 0.538 | 1.185 | 0.0053 |
| | 6 | 0.39 | 0.28 | -0.599 | 0.342 | 0.053 | 2.34 |

## Script Delphi

```
unit Unit31;

interface
uses  sysutils;

const

        max_units = 35;
        max_order = 10;
        max_curve_points = 10;
        max_period = 24;
        max_total_segments = 200;
        total_gen_tolerance = 0.01 ;
        ihr_tolerance = 0.000001 ;


        alpha : real = 1; {See note in loss matrix procedure}

type

        unit_array_real = array[1..max_units] of real;
        unit_name_array = array[1..max_units] of string;
        coefficients = array[0..max_order] of real;
        unit_poly_array = array[1..max_units] of coefficients;
        curve_points = array[0..max_curve_points] of real;
        unit_crve_array = array[1..max_units] of curve_points;
        system_ihr_array_real = array[1..max_total_segments] of real;
        system_ihr_array_integer = array[1..max_total_segments] of integer;
        B_matrix = array[1..max_units] of unit_array_real;
        filename_array = string;
        curvetype_list = (poly,pinc,pio);
        losstype_list = (noloss,constpf,lossform);
        solution_type_list = (lamsearch,tbllookup);
        schedtype_list = (totgen,totload);
        unit_array_period= array[1..max_period] of unit_array_real;
        period_array_real= array[1..max_period] of real;


var
       ioerr : boolean;
        ioval : integer;
      data_period:unit_array_period;     {Generation each unit on each period}
      total_period:period_array_real;    {total cost each period}
      genname:unit_name_array;           {Generator name identifier}
      p:unit_array_real;                 {Present value of P}
      pmin:unit_array_real;              {Minimum MW}
      pmax:unit_array_real;              {Maximum MW}
      UR:unit_array_real;
      DR:unit_array_real;
      maxup:unit_array_real;
      maxdown:unit_array_real;
      unitsebelum:unit_array_real;
      ramprate:boolean;
      minihr:unit_array_real;            {Minimum unit incremental heat rate}
      maxihr:unit_array_real;            {Maximum unit incremental heat rate}
      fuelcost:unit_array_real;          {Fuel cost ( $/fuel unit )}
      coeff : unit_poly_array;           {Unit polynomial coefficients}
      ihr_mwpoint:unit_curve_array;        {MW points on ihr cost curve}
      ihr_cost:unit_curve_array;           {Cost points for ihr curve}
      io_mwpoint : unit_curve_array;     {MW Points on unit io curve}
      io_cost : unit_curve_array;        {Cost points on io curve}
      minput:unit_array_real;            {Minimum input for PINC curves }
      penfac:unit_array_real;            {Loss penalty factor}
      penfacD:unit_array_real;            {Loss penalty factor}
      unitmin:unit_array_real;            {Initial Minimum Unit Generation}
      unitmax:unit_array_real;            {Initial Maximum Unit Generaion}
      totalcost:real;
      loadjam:unit_array_real;

      b00:real;                          {Loss matrix constant}
      b0:unit_array_real;                {Loss matrix linear terms}
      b:B_matrix;                        {Loss matrix quadratic terms}
      seginccost:system_ihr_array_real;  {Segment inc cost for table look up }
      segunit:system_ihr_array_integer;  {Unit associated with segment}
      segmw:system_ihr_array_real;       {MW contributed by segment}
      order:system_ihr_array_integer;    {Order routine output}
      ordvalue:system_ihr_array_real;    {Numbers to be ordered}
      inputfile:text;
      filename:filename_array;
      title1, title2 : string[80];
      print_output, diagflag, read_data : boolean;
      inputchar,quitflag : char;
      linenumber, ngen : integer;
      mwlosses, lambda : real;
      curvetype_input, losstype_input : string[8];
      number_string : string[20];
      curveorder : integer;
```

```
    curvetype : curvetype_list;
    losstype : losstype_list;
    solution_type : solution_type_list;
    schedtype : schedtype_list;
    pgenmax, pgenmin, SRGenTotal : real;
    FF:Text;
    NoGenerator:integer ;
    NomerOrder:integer;
    ModeDataPembangkit:integer;
    ProsesRun:boolean;
    SR : real;
    numper : integer;
    jamstart : real;
procedure datainput(namafile :string);
procedure ihr_ftn(i : integer; unitmw : real;  var unitihr : real );
procedure GetFileName(s :string;var d:string;var f:string;var e :string);
procedure datadump( loadjam:real;unitsebelum:unit_array_real; var outfile:text );
procedure lambda_search_dispatch( schedmw:real; var lambda : real);
procedure loss_matrix_ftn;
procedure inverse_ihr_ftn(i : integer; unitihr : real; var unitmw : real );

procedure order_routine(        numorder : integer;
                                ordertable : system_ihr_array_real;
                        var     orderindex : system_ihr_array_integer );
procedure output_routine( schedmw:real; var outfile : text; lambda : real );
procedure prod_cost(        i : integer;
                        unitmw : real;
                var   unitcost : real );
procedure dataOutput(namafile :string);
PROCEDURE output_final( VAR  OUTFILE : TEXT );

function cekIoPoint(var unitG,Order:integer):boolean;
function cekIhrPoint(var unitG,Order:integer):boolean;
function CekEdcFile(filename:string): boolean;


implementation
function CekEdcFile(filename:string): boolean;
label keluar;
var s,d,f,e:string;
ff:text;
bc : boolean;
begin
bc:=false;
getfilename(filename,d,f,e);
 s:=trim(uppercase(f))+'.'+uppercase(e);
if s = 'EDC1.DAT' then bc:= true;
if s = 'EDCTEST.DAT' then bc:= true;

if s = 'EDC2.DAT' then bc:= true;
if s = 'EDC3.DAT'  then bc:= true;
if s = 'EDC4.DAT'  then bc:= true;
if s = 'EDC5.DAT'  then bc:= true;

if s = 'EXDED.DAT'  then bc:= true;
if s = 'EX3B.DAT'  then bc:= true;
if s = 'EX3C.DAT'  then bc:= true;
if s = 'EX3D.DAT'  then bc:= true;

if s = 'PR32.DAT'  then bc:= true;
if s = 'PR33.DAT'  then bc:= true;
if s = 'PR38.DAT'   then bc:= true;
if s = 'PR43A.DAT'  then bc:= true;
if s = 'PR43B.DAT'   then bc:= true;
if s = 'EX4D.DAT'   then bc:= true;

if bc = true then goto Keluar;

assign(ff,filename);
reset(ff);
readln(ff,s);
if pos('EDC FILE >>',s)>0 then bc:=true;
close(ff);


keluar:
cekedcfile:=bc;
end;
function cekIoPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIopoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (io_mwpoint[i,j+1] - io_mwpoint[i,j])<=0 then
      begin
        cekIoPoint:=false;
        UnitG:=i;
```

```
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;


function cekIhrPoint(var unitG,Order:integer):boolean;
label keluar;
var i,j:integer;
begin
  cekIhrpoint:=true;
  for i:=1 to ngen do
  begin
    for j:= 0 to curveorder-1 do begin
      if (ihr_mwpoint[i,j+1] - ihr_mwpoint[i,j])<=0 then
      begin
        cekIhrPoint:=false;
        UnitG:=i;
        Order:=j;
        goto keluar;
      end;
    end;
  end;
keluar:
end;

procedure prod_cost(       i : integer;
                       unitmw : real;
             var   unitcost : real );

      { Routine to return unit production cost given unit output in mw}
      { input : unit index = i}
      {         unit MW =  unitmw}
      { output: unit production cost = unitcost}

var
       j : integer;
       partmw, unitihr, segmentcost : real;

label return;

begin

case curvetype of
  poly :                          {Polynomial I/O curve}
      begin
      unitcost := 0;
      for j := curveorder downto 1 do
      unitcost := ( unitcost + coeff[ i,j ] ) * unitmw;

      unitcost := unitcost + coeff[ i,0 ];
      unitcost := unitcost * fuelcost[ i ];
      goto return
      end;

  pinc :                          {Piecewize incremental curve}
      begin
      unitcost := minput[ i ] * fuelcost[ i ];
      for j := 1 to curveorder do
         begin
         if (unitmw > ihr_mwpoint[ i,j ]) and ( j < curveorder ) then
                         {Calculate area under complete segment}
            begin
            segmentcost :=  ( (ihr_cost[i,j] + ihr_cost[i,j-1] )/2.0 ) *
                            ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) *
                                    fuelcost[ i ];
            unitcost := unitcost + segmentcost;
            end
            else
                         {Calculate area under partial segment}
            begin
            partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                              (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
            unitihr := ihr_cost[i,j-1] +
                           ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw;
            segmentcost :=  ( (unitihr + ihr_cost[i,j-1])/ 2.0 ) *
                             ( unitmw - ihr_mwpoint[i,j-1] )  *
                             fuelcost[ i ];
            unitcost := unitcost + segmentcost;

            goto return;
            end;
         end;

      end;

   pio :                          {Piecewize I/O curve}
```

```
            begin
            for j := 1 to curveorder do
               begin
               if io_mwpoint[i,j] > unitmw then
                  begin
                  partmw := (unitmw-io_mwpoint[i,j-1] ) /
                                    (io_mwpoint[i,j]-io_mwpoint[i,j-1] );
                  unitcost := io_cost[i,j-1] +
                                    ( io_cost[i,j]-io_cost[i,j-1] ) * partmw;
                  unitcost := unitcost * fuelcost[ i ];
                  goto return
                  end;
               if j = curveorder then     {Unit is at or above pmax}
                  begin
                  unitcost := io_cost[ i, j ] * fuelcost[ i ];
                  goto return
                  end;
               end;
            end;
         end;  { End of case statement}

      return:

      end; { End procedure }

      procedure output_routine( schedmw:real; var outfile : text; lambda : real );

      var
            limittxt : string[5];
            totalgen, totalload, totalmaxdown, totalmaxup : real;
            unitihr, unitinccost, unitcost : real;
            i : integer;

      label return;

      begin

      writeln(outfile);
      writeln(outfile,
            'generator  output    limit    maxup    maxdown     inc cost penalty fact  operating
cost');
      writeln(outfile,
            '                      mw                mw        mw         $/mwhr                    $/hr
');
      writeln(outfile,
            '---------  ------   -----  -------- --------  ---------- ----------   ------------
-');

      totalgen := 0.0;
      totalcost := 0.0;
      totalmaxup:=0;
      totalmaxdown:=0;


      for i := 1 to ngen do
         begin
         write(outfile,genname[i]:9);
         write(outfile, ' ',p[i]:6:1, '    ');
         limittxt :=  '     ';
         if abs( p[i] - pmin[i] ) < total_gen_tolerance then limittxt := 'min  ';
         if abs( p[i] - pmax[i] ) < total_gen_tolerance then limittxt := 'max  ' ;
         write(outfile, limittxt );

         ihr_ftn( i, p[ i ], unitihr) ;    {Get unit incremental heat rate}

         unitinccost := unitihr * fuelcost[i];
         write(outfile, maxup[i]:10:3,'');
         write(outfile, maxdown[i]:10:3, '');
         write(outfile, unitinccost:9:4);
         write(outfile, '   ',penfac[i]:9:4, '    ');

         prod_cost( i, p[ i ], unitcost );          {Calculate unit operating cost}

         writeln(outfile, '   ',unitcost:9:2);
         totalgen := totalgen + p[i];
         totalcost := totalcost + unitcost;
         totalmaxup:= totalmaxup + maxup[i];
         totalmaxdown:= totalmaxdown + maxdown[i];
         end;
      writeln(outfile,
            '---------  ------                                                                  ---------
-----');
      write(outfile, ' totals');
      write(outfile, totalgen:9:1,                                        ', totalcost:9:2);

      writeln(outfile);
      writeln(outfile,' lambda = ', lambda:10:4 );
      writeln(outfile);

      if (schedtype = totgen ) and ( losstype <> lossform ) then goto return;
```

```
if schedtype=totload then totalload := schedmw;

if schedtype=totgen then  totalload := totalgen - mwlosses;

writeln(outfile, 'total load = ',totalload:10:1,
                 '  total losses = ',mwlosses:10:1);
writeln(outfile);
writeln(outfile, 'maxup for next periode = ',totalmaxup:10:3);
writeln(outfile);
writeln(outfile, 'maxdown for next periode = ',totalmaxdown:10:3);


return:

end; { End procedure }

PROCEDURE output_final( VAR  OUTFILE : TEXT );

VAR I,J : INTEGER;

BEGIN


      WRITELN(OUTFILE);

      WRITELN(OUTFILE,'FINAL OUTPUT DYNAMIC ECONOMIC DISPATCH ');
      WRITELN (OUTFILE);
      WRITELN(OUTFILE,'NOTE:(PERIOD 1 NOT ALWAYS CALCULATE FOR LOAD AT HOUR 1)');
      WRITELN (OUTFILE);
      WRITE(OUTFILE,'PERIOD       UNIT STATUS ');
      FOR I:=1 TO ngen*8-11 DO WRITE(OUTFILE,' ');
      WRITELN(OUTFILE,' PCOST     LOAD');
      WRITE(OUTFILE,'       ');
      FOR J := 1 TO ngen DO WRITE(OUTFILE,J:8);
      WRITELN(OUTFILE,'          R/HR      MW');
      FOR J := 1 TO 34+ngen*8 DO WRITE(OUTFILE,'-');
      WRITELN(OUTFILE);
      FOR i := 1 TO numper DO
      BEGIN
        WRITE(OUTFILE,i:4,'    ');
        FOR J := 1 TO ngen DO
          WRITE(OUTFILE,' ',data_period[i,j]:6:1,' ');
        WRITELN(OUTFILE,' ',total_period[i]:9:2,' ',loadjam[i]:8:1);
        {OPTSTATE := PATH[ OPTSTATE ];   }
      END;

END;


procedure order_routine(        numorder : integer;
                                ordertable : system_ihr_array_real;
                          var   orderindex : system_ihr_array_integer );

{ subroutine to order a list, least first                                  }
{                                                                          }
{ input numorder = the number of items to be ordered                       }
{ input ordertable = the items to be ordered                               }
{ output orderindex = pointer to order value table                         }
{                                                                          }
{ nxt = Table used in order subroutine                                     }
{                                                                          }
  var
      stop:boolean;
      i,j,top,last,indx:integer;
      nxt : system_ihr_array_integer;

  begin
    for i := 1 to numorder do begin
      if (i <= 1) then begin
        top := 1;
        nxt[ 1 ] := 0;
        end
      else begin
        j := top;
        last := 0;
        repeat
          stop := true;
          if (ordertable[ i ] > ordertable[ j ]) then begin
            last := j;
            j := nxt[ j ];
            stop  := (j = 0);
            if (stop) then begin
              nxt[ last ] := i;
              nxt[ i ] := 0;
            end
            end
          else begin
            if (j <> top) then begin
```

```
              nxt[ last ] := i;            { j not = top }
              nxt[ i ] := j;
              end
            else begin
              top := i;                    { j = top }
              nxt[ i ] := j;
            end;
          end;
        until stop;
      end;
    end;
    indx := 1;
    j := top;
    repeat
      orderindex[ indx ] := j;
      j := nxt[ j ];
      indx := indx + 1;
    until (j = 0);
  end;


procedure inverse_ihr_ftn(       i : integer;
                           unitihr : real;
                       var unitmw : real );

      { Routine to return unit MW given unit incremental heat rate}
      { input : unit number = i}
      {         unit inc heat rate = unitihr}
      { output: unit MW stored in unitmw}

label  return;

var
       unitihr1, delihr, partihr, dihrdp : real;
       segmentihr : real;
       j, step : integer;

begin

if unitihr >= maxihr[ i ] then
    begin
    unitmw := pmax[ i ];
    goto return
    end;
if unitihr <= minihr[ i ] then
    begin
    unitmw := pmin[ i ];
    goto return
    end;

case curvetype of
  poly :                        {Polynomial curve}
      begin
      if curveorder <= 1 then
       begin
       if unitihr > coeff[ i,1 ] then unitmw:=pmax[i] else unitmw:=pmin[i];
       goto return
       end;

      if curveorder = 2 then
       begin
       unitmw := ( unitihr - coeff[ i,1 ] ) / ( 2.0 * coeff[ i,2 ] );
       goto return
       end;

            { for curves of order >= 3 search for unitmw using Newtons method }

      unitmw := ( pmin[ i ] + pmax[ i ] )/ 2.0;
      step := 0;
      repeat
      step := step + 1;

      unitihr1 := 0;                    {Calc unitihr at unitmw as unitihr1}
      for j := curveorder downto 2 do
         unitihr1 := ( unitihr1 + j * coeff[i,j] ) * unitmw;
      unitihr1 := unitihr1 + coeff[i,1];
      delihr := unitihr - unitihr1;
      if abs( delihr ) < ihr_tolerance then goto return;

      dihrdp := 0;                      {Calc curve second derivative}
      for j := curveorder downto 3 do
         dihrdp := ( dihrdp + j*(j-1) * coeff[i,j] ) * unitmw;
      dihrdp := dihrdp + 2.0 * coeff[ i,2 ];
      unitmw := unitmw + delihr/dihrdp;

      until step > 35;

      goto return
      end;
```

```
  pinc :                              {Piecewize incremental curve}
      begin
      j := 0 ;
      repeat
      j := j + 1;
      until (ihr_cost[i,j] > unitihr) or
            (j = curveorder);

      partihr := ( unitihr - ihr_cost[i,j-1] )/
                           ( ihr_cost[i,j] - ihr_cost[i,j-1] );
      unitmw := ihr_mwpoint[i,j-1] +
                    ( ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] ) * partihr;
      goto return
      end;

   pio :                              {Piecewise I/O curve}
      begin
      j := 0;
      repeat
      j := j + 1;
      if j = curveorder then
         begin
         unitmw := io_mwpoint[i,j];
         goto return
         end;
      segmentihr :=(io_cost[i,j] - io_cost[i,j-1]) /
                              (io_mwpoint[i,j] - io_mwpoint[i,j-1]);
      until segmentihr >= unitihr;

      unitmw := io_mwpoint[ i,j-1 ];
      goto return
      end;

   end;  { End of case statement}

return:

end; { End procedure }

procedure loss_matrix_ftn;

      { Routine to calculate losses and penalty factors from loss formula}
      {  Input:   Table of unit generation p[ i ]}
      {           Loss formula b[ i,j ], b0[ i ], b00}
      {  Output:  losses mwlosses and penalty factors penfac[ i ]}
      {  Note: loss formula expects p(i)'s to be in per unit so divide by 100.}

var
        i, j : integer;
        incloss, penfac_old, penfac_new : real;

label return;

begin

      mwlosses := b00;
      for i := 1 to ngen do
         begin
         mwlosses := mwlosses + b0[i] *
                           ( p[i]/100.0) + b[i,i] * sqr( p[i]/100.0 );
         for j := i+1 to ngen do
           mwlosses := mwlosses + 2.0 * ( p[i]/100.0) * ( p[j]/100.0 ) * b[i,j]
         end;

      mwlosses := mwlosses * 100.0;


      for i := 1 to ngen do
         begin
         penfac_old := penfac[ i ];
         incloss := b0[i];
         for j := 1 to ngen do
            incloss := incloss + 2.0 * ( p[j]/100.0 ) * b[i,j];
         penfac_new := 1.0 / ( 1.0 - incloss );
         penfac[ i ] := penfac_old + alpha *
                                          ( penfac_new  -  penfac_old)
         end;

{ Note, in the formula above the penalty factor is "filtered" by the }
{ alpha filtering constant. If alpha is set to 1.0 no filtering action }
{ takes place, if alpha is 0.0 penfac is constant at 1.0 , suggested }
{ value for alpha is 0.5 to 0.9 }

return:

end; { End procedure }


procedure lambda_search_dispatch( schedmw:real; var lambda : real);
```

```
        var
                i, n, lossiter : integer;
                lambdamin, lambdamax : real;
                lambdastart, deltalambda, targetgen : real;
                unitihr, unitmw, totalgen, totalmaxup, totalmaxdown :  real;
                endloop : boolean;

        begin

        for i := 1 to ngen do                    { Set unit output to midrange}
          begin
          p[ i ] := ( pmin[ i ] + pmax[ i ] ) / 2.0
          end;

        lossiter := 0;
        endloop := false;

        repeat                                   {Top of iterative loop with losses}

          lambdamin := 10000.0;
          lambdamax := 0.0;
          mwlosses := 0 ;
          if losstype = lossform then   { Calc losses and pen factors}
          begin
            loss_matrix_ftn;
            writeln(ff);
            writeln(ff,' mw losses = ',mwlosses:10:1);
          end;

          for i := 1 to ngen do                  {Calculate max and min lambdas}
            begin
            ihr_ftn (i,pmax[i],maxihr[i]);
            lambda := maxihr[ i ] * penfac[ i ] * fuelcost[ i ];
            if lambda > lambdamax then lambdamax := lambda;
            ihr_ftn (i,pmin[i],minihr[i]);
            lambda := minihr[ i ] * penfac[ i ] * fuelcost[ i ];
            if lambda < lambdamin then lambdamin := lambda
            end;


            writeln(ff,' lambda limits = ',lambdamin:10:4,lambdamax:10:4);


          lambdastart := ( lambdamax + lambdamin ) / 2.0;
          deltalambda := ( lambdamax - lambdamin ) / 2.0;


           writeln(ff,' lambdastart deltalambda = ',lambdastart:10:4,deltalambda:10:4);
          {Set up total generation target}
          if schedtype = totgen then targetgen := schedmw;
          if schedtype = totload then targetgen := schedmw + mwlosses;
          {Lambda search}
          lambda := lambdastart;
          writeln(ff, ' targetgen = ',targetgen:10:1);

          n := 0;
          repeat                           {Top of lambda search loop}
              n := n + 1;
              totalgen := 0;
              totalmaxup:=0;
              totalmaxdown:=0;

              for i := 1 to ngen do
                begin
                unitihr := lambda / ( penfac[ i ] * fuelcost[ i ] )  ;
                inverse_ihr_ftn( i, unitihr, unitmw );  {For given unitihr get unitmw}
                p[ i ] := unitmw;
                maxup[i]:=p[i]+UR[i];
                  if maxup[i]>unitmax[i] then maxup[i]:=(unitmax[i]-p[i]) else maxup[i]:=UR[i] ;
                maxdown[i]:=p[i]-DR[i];
                  if     maxdown[i]<unitmin[i]     then     maxdown[i]:=(p[i]-unitmin[i])     else
maxdown[i]:=DR[i];
                totalgen := totalgen + p[ i ];
                totalmaxup:=totalmaxup + UR[i];
                totalmaxdown:=totalmaxdown + DR[i];

                end;


                  writeln(ff,' lambda = ',lambda:2:4,'  totalgen = ',totalgen:10:3,' delta lambda
= ',deltalambda:1:4);


              if abs( totalgen - targetgen ) >= total_gen_tolerance then
                begin
                if totalgen > targetgen then lambda := lambda - deltalambda;
                if totalgen < targetgen then lambda := lambda + deltalambda;
                deltalambda := deltalambda / 2.0
```

```
        end;

      until ( abs( totalgen - targetgen ) < total_gen_tolerance ) or
            ( n > 35 ) ;

{See if another loss iteration is needed}

      if losstype <> lossform then endloop := true;
      lossiter := lossiter + 1;
      if lossiter > 10 then endloop := true ;
      if  abs( totalgen - targetgen ) > total_gen_tolerance  then  ProsesRun:=false;
      //else ProsesRun:=True;        //check if the program really get the right result

until endloop;


end; { End Lambda search procedure }

procedure GetFileName(s :string;var d:string;var f:string;var e :string);
var ss ,sd:string;
  n:integer;
begin
  ss := s;
  d := '';
  n :=  pos('\',ss);
  while  n>0 do
  begin
     sd := copy(ss,1,n);
     d := d+sd;
     delete(ss,1,n);
  n :=  pos('\',ss);
  end;
  n := pos('.',ss);
  if n = 0 then
  begin
     f := ss;
     e:= '';
  end else
  begin
     f:=copy(ss,1,n-1);
     delete(ss,1,n);
     e := ss;
  end;
end;
procedure IOCheck( linenumber : integer );
begin
  IOval := IOresult;
  IOErr := (IOVal <> 0);
end; { of proc IOCheck }


procedure datainput(namafile :string);

label quit;

var i,j,k,jj : integer;
    a : char;

begin
print_output := True;

linenumber := 0;

assign(inputfile, namafile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                    }
{ Open data file     }
{                    }
reset(inputfile);
     iocheck( linenumber );
     if ioerr then goto quit;
{                    }
{ Read file header }
{                    }
linenumber := linenumber + 1;
readln( inputfile, title1 );
     iocheck( linenumber );
     if ioerr then goto quit;
linenumber := linenumber + 1;
readln( inputfile, title2 );
     iocheck( linenumber );
     if ioerr then goto quit;

     {                                                          }
     {  Read Number of generators, curve type, loss type  }
     {                                                          }
linenumber := linenumber + 1;
read( inputfile, ngen );
     iocheck( linenumber );
```

```
     if ioerr then goto quit;

repeat
read( inputfile, inputchar );
       iocheck( linenumber );
       if ioerr then goto quit;
until inputchar <> ' ';
curvetype_input := inputchar;
repeat
read( inputfile, inputchar );
      iocheck( linenumber );
      if ioerr then goto quit;
if inputchar <>' ' then curvetype_input := curvetype_input + inputchar;
until inputchar = ' ';

read( inputfile, curveorder );
      iocheck( linenumber );
      if ioerr then goto quit;

repeat
read(inputfile, inputchar );
     iocheck( linenumber );
     if ioerr then goto quit;
until inputchar <>  ';
losstype_input := inputchar;
readln(inputfile);

 {                                                    }
 {  Set up internal variables for curvetype and losstype  }
 {                                                    }

if (curvetype_input = 'poly') or
   (curvetype_input = 'POLY') then curvetype := poly;

if (curvetype_input = 'pinc') or
   (curvetype_input = 'PINC') then curvetype := pinc;

if (curvetype_input = 'pio' ) or
   (curvetype_input = 'PIO' ) then curvetype := pio;

if (losstype_input = 'n'  ) or
   (losstype_input = 'N'  ) then losstype := noloss;

if (losstype_input = 'c'  ) or
   (losstype_input = 'C'  ) then losstype := constpf;

if (losstype_input = 'l'  ) or
   (losstype_input = 'L'  ) then losstype := lossform;


          {                          }
          { Read generator data }
          {                          }

for i := 1 to ngen do
   begin  { Read generator name }
       linenumber := linenumber + 1;
       repeat
       read( inputfile, inputchar );
             iocheck( linenumber );
             if ioerr then goto quit;
       until inputchar <>  ';
       genname[i] := inputchar;
       repeat
       read( inputfile, inputchar );
             iocheck( linenumber );
             if ioerr then goto quit;
       if inputchar <>' ' then genname[i] := genname[i] + inputchar;
       until inputchar = ' ';
          {                                  }
          { Read generator min, max, fuelcost }
          {                                  }
       readln( inputfile, pmin[i], pmax[i], fuelcost[i], DR[i], UR[i] );
             iocheck( linenumber );
             if ioerr then goto quit;
          {                          }
          { Read generator cost curve data    }
          {                          }
       case curvetype of
         poly :
                begin  { read polynomial curve data}
                for j := 0 to curveorder do
                  begin
                    linenumber := linenumber + 1;
                    readln( inputfile, coeff[i,j] );
                     iocheck( linenumber );
                     if ioerr then goto quit;
                  end;
                end;
```

```
            pinc :
                  begin  { read piecewise incremental cost curve data}
                  linenumber := linenumber + 1;
                  readln( inputfile, minput[i] );
                    iocheck( linenumber );
                    if ioerr then goto quit;
                  for j := 0 to curveorder  do
                    begin
                      linenumber := linenumber + 1;
                      readln( inputfile, ihr_mwpoint[i,j], ihr_cost[i,j] );
                        iocheck( linenumber );
                        if ioerr then goto quit;
                    end;
                  end;

            pio  :
                  begin  { read piecewise I/O curve data}
                  for j := 0 to curveorder do
                    begin
                      linenumber := linenumber + 1;
                      readln( inputfile, io_mwpoint[i,j], io_cost[i,j] );
                        iocheck( linenumber );
                        if ioerr then goto quit;
                    end;
                  end;

            end;  { End of case statement}
      end;
            {                   }
            { Read loss data }
            {                   }

    case losstype of

      noloss :
            begin
            for i := 1 to ngen do      { Init penalty factors}
                penfac[ i ] := 1.0
            end;

      constpf  :
          begin   { read constant penalty factor data}
            linenumber := linenumber + 1;
            for i := 1 to  ngen do
              begin
              if i < ngen then
              read( inputfile, penfac[i] )
              else
              readln( inputfile, penfac[ngen] );
                 iocheck( linenumber );
                 if ioerr then goto quit
              end;
          end;

      lossform :

          begin  { read loss formula data}

          linenumber := linenumber + 1;
          readln( inputfile, b00 );
                iocheck( linenumber );
                if ioerr then goto quit;

          linenumber := linenumber + 1;
          for j := 1 to  ngen  do
            begin
            if j < ngen then
               read( inputfile, b0[ j ] )
               else
               readln( inputfile, b0[ngen] );
                iocheck( linenumber );
                if ioerr then goto quit;
            end;

          for i := 1 to ngen do
            begin
            linenumber := linenumber + 1;
            for j := 1 to ngen  do
              begin
              if j < ngen then
                  read( inputfile, b[ i, j ] )
                  else
                  readln( inputfile, b[ i, ngen ] );
                  iocheck( linenumber );
                  if ioerr then goto quit
              end;
            end;
            for i := 1 to ngen do      { Init penalty factors}
                penfac[ i ] := 1.0
          end;
```

```
      end;  { End of case statement}
      {                                }
      {  End of data input, close file  }
      {                                }
      close ( inputfile );

      {A very useful table is the incremental cost at the max and min of each unit. }
      {These are calculated and stored in tables maxihr and minihr.}
      {Also calculate the max and min generation}


      quit:

      end;   { end of data input }



      procedure dataOutput(namafile :string);
      var i,j,k,jj : integer;
          a : char;
          d,e,f:string;
         var inputfile:text;
      begin
      getfilename(namafile,d,f,e);
      assign(inputfile, namafile);
      rewrite(inputfile);
      writeLn( inputfile, 'EDC FILE >> '+f+'.'+e );
      writeln( inputfile, '====================================');
      write( inputfile, ngen );
      write(inputfile,' ');

      if  curvetype = poly then write(inputfile,'POLY');
      if  curvetype = pinc then write(inputfile,'PINC');
      if  curvetype = PIO then write(inputfile,'PIO');
      write(inputfile,' ');
      write(inputfile,curveorder);
      write(inputfile,' ');

      if losstype = noloss  then write(inputfile,'NOLOSS');
      if losstype = constpf  then write(inputfile,'CONSTPF');
      if losstype = lossform  then write(inputfile,'LOSSFORM');
      writeln(inputfile);
      for i := 1 to ngen do
          begin
            write(inputfile,genname[i]);
            write(inputfile,' ');
            writeln(  inputfile,  pmin[i]:15:6,  pmax[i]:15:6,  fuelcost[i]:15:6,  DR[i]:15:6,
UR[i]:15:6 );
           case curvetype of
               poly :
                      begin
                      for j := 0 to curveorder do
                        begin
                          writeln( inputfile, coeff[i,j]:15:6 );
                        end;
                      end;

               pinc :
                      begin
                      writeln( inputfile, minput[i]:15:6 );
                      for j := 0 to curveorder  do
                        begin
                          writeln( inputfile, ihr_mwpoint[i,j]:15:6, ihr_cost[i,j]:15:6 );
                        end;
                      end;

               pio  :
                      begin
                      for j := 0 to curveorder do
                        begin
                          writeln( inputfile, io_mwpoint[i,j]:15:6, io_cost[i,j]:15:6);
                        end;
                      end;

               end;
          end;

      case losstype of

        noloss :
              begin
              for i := 1 to ngen do     { Init penalty factors}
                penfac[ i ] := 1.0
              end;

        constpf  :
```

```
        begin
          for i := 1 to  ngen do
          begin
          if i < ngen then
          begin
          write( inputfile, penfac[i]);
          write(inputfile,' ');

          end
          else writeln( inputfile, penfac[ngen]);
          end;
        end;

    lossform :

        begin

        linenumber := linenumber + 1;
        writeln( inputfile, b00);
         for j := 1 to  ngen  do
          begin
          if j < ngen then
             begin
             write( inputfile, b0[ j ]);
             write(inputfile,' ');
             end
             else
             writeln( inputfile, b0[ngen]);
          end;

        for i := 1 to ngen do
          begin
          for j := 1 to ngen  do
           begin
           if j < ngen then
              begin
              write( inputfile, b[i, j]);
              write(inputfile,' ');

              end
              else
              writeln( inputfile, b[i,ngen]);
           end;
          end;

          for i := 1 to ngen do      { Init penalty factors}
                penfac[ i ] := 1.0

        end;

    end;


close ( inputfile );


end;

procedure ihr_ftn(          i : integer;
                      unitmw : real;
                  var unitihr : real );

        { Routine to return unit incremental heat rate given unit output in MW }
        { input : unit index = i}
        {         unit mw = unitmw}
        { output: unit incremental heat rate = unitihr}

var
          partmw : real;
          j : integer;

begin

case curvetype of
  poly :                    {Polynomial I/O curve}
        begin
        unitihr := 0.0;
        for j := curveorder downto 2 do
          unitihr := ( unitihr + j * coeff[ i,j ] ) * unitmw;
        unitihr := unitihr + coeff[ i,1 ]
        end;

  pinc :                    {Piecewize incremental curve}
        begin
        j := 0;
        repeat
        j := j + 1;
        until (ihr_mwpoint[ i,j ] > unitmw) or (j = curveorder);
```

```
            partmw := (unitmw - ihr_mwpoint[i,j-1] )/
                              (ihr_mwpoint[i,j] - ihr_mwpoint[i,j-1] );
            unitihr := ihr_cost[i,j-1] + ( ihr_cost[i,j] - ihr_cost[i,j-1] ) * partmw
            end;

        pio  :                      {Piecewize I/O curve}
            begin
            j := 0;
            repeat
            j := j + 1;
            until (io_mwpoint[ i,j ] > unitmw) or (j = curveorder);

            unitihr := (io_cost[i,j] - io_cost[i,j-1] ) /
                              ( io_mwpoint[i,j] - io_mwpoint[i,j-1] )
            end;
        end;  { End of case statement}


    end; { End ihr_ftn procedure }

    procedure datadump( loadjam:real; unitsebelum:unit_array_real ;var outfile:text );


    var
       i,j: integer;

    begin
    //if jamstart < 1 then jamstart:=1;
    for i := 1 to ngen do
    begin
    pmax[i]:=unitmax[i]-(SR/100)*unitmax[i];
    pmin[i]:=unitmin[i];
    if unitsebelum[i] <> 0 then if ramprate then
      begin
      if     (unitsebelum[i]+UR[i])<(unitmax[i]-(SR/100)*unitmax[i])     then     pmax[i]     :=
unitsebelum[i]+UR[i];
      if (unitsebelum[i]-DR[i])>(unitmin[i]) then pmin[i] := unitsebelum[i]-DR[i];
      end;
    end;

    writeln(outfile);
    writeln(outfile, title1);
    writeln(outfile, title2);
    writeln(outfile);
    writeln(outfile ,' number of generator units = ',ngen );

    case curvetype of
      poly : writeln(outfile ,' unit curve type = poly ');
      pinc : writeln(outfile ,' unit curve type = pinc ');
      pio  : writeln(outfile ,' unit curve type = pio');
      end;  { End of case statement}

    writeln(outfile ,' curve order = ',curveorder);

    case losstype  of
      noloss :   writeln(outfile ,' network loss representation = noloss ');
      constpf :  writeln(outfile ,' network loss representation = constpf ');
      lossform : writeln(outfile ,' network loss representation = lossform ');
      end;  { End of case statement}
    for i := 1 to ngen do
      begin
       writeln(outfile);
       write(outfile, genname[i],' limits = ',pmin[i]:7:2, ' ',pmax[i]:7:2 );
       writeln(outfile, ' fuelcost = ',fuelcost[i]:10:4 );
         case curvetype of
              poly :
                  begin
                  writeln(outfile,'  polynomial coefficients' );
                  for j := 0 to curveorder do
                    begin
                    writeln(outfile, coeff[i,j]:15:6);
                    end;
                  end;
              pinc :
                  begin
                  writeln(outfile,' incremental cost curve points');
                  writeln(outfile,'input at pmin = ',minput[i]:10:2);
                  for j := 0 to curveorder do
                    begin
                    writeln(outfile,ihr_mwpoint[i,j]:9:2,ihr_cost[i,j]:9:3 )
                    end;
                  writeln(outfile);
                  end;
              pio :
                  begin
                  writeln(outfile,' cost curve points');
                  for j := 0 to curveorder do
                    begin
                    writeln(outfile,io_mwpoint[i,j]:9:2,'  ',io_cost[i,j]:9:3 )
                    end;
```

# DAFTAR GAMBAR

# DAFTAR PUSTAKA


[1] Benhima, F, "*Solving Dynamic Economic Load Dispatch Witg Ramp rate Limit Using Quadratic Programming*", IEEE 978-1-4799-1255-1/13, 2013.

[2] Rabiee, Abbas, "*Fast Dynamic Economic Power Dispatch Problems Solution Via Optimality Condition Decomposition*", IEEE Transaction on Power System, Vol. 29, No. 2, March 2014.

[3] Allen J. Wood, Bruce F. Wollenberg, "Power, Generation, Operation, and Control", John Willey & Sons Inc, America, 1996.

[4] Kusnassriyanto, "*Belajar Pemrograman Delphi",* MODULA, Bandung, 2011.

[5] Saadat, Hadi, "*Power System Analysis 2nd Edition",* McGrowHill, Ch.1, 1999.

[6] Caelho, Leandro and Lee,Chu-Sheng, "*Solving Economic Load Dispatch Problem in Power Systems Using Chaotic and Gaussian Particle Swarm Optimization Approaches*", ELSEVIER Electrical Power and Energy Systems, Vol. 30, pp. 297-307, 2008.

[7] El-Harawary, M. E, "Electrical Power System: Design and Analysis-Rev Printing", IEEE Press Power System Enginering Series, United States of America, 1995.

[8] Mahatmya, Atya, "Implementasi Algoritma Ant Colony Optimization untuk Menyelesaikan Persamalahan Dynamic Economic Dispatch dengan Memperhatikan Rugi-rugi Daya Transmis dan Valve Point Effect", Jurusan Teknik Elektro FTI-ITS, Surabaya, 2013.

[9] Benhima, F, "Constrained Dynamic Economical Dispatch Using a Compact Quadratic Programming Method Including Losses", The International Conference on Electronics and Oil: From Theory to Aplication, Algeria, March 2013.

[10] Kumar, Pardeeep, "*Dynamic Economic Dispatch Using Defferentila Evolution Algorithm",* Department of Electrical and Instrumentation Enginering Thapar University, Punjab, 2013.

*Halaman ini sengaja dikosongkan*

# RIWAYAT HIDUP PENULIS

**HARDI RIZKYANTO,** lahir di Bojonegoro, 26 Juli 1993. Penulis tamat dari bangku sekolah dasar di SDN Simomulyo 4 Surabaya tahun 2005 dan melanjutkan di sekolah menengah pertama di SMP Al-Hikmah Surabaya, lulus tahun 2008. Setelah lulus SMP, penulis melanjutkan sekolah ke SMAN 9 Surabaya. Pada tahun 2011, penulis melanjutkan studi S1 di Institut Teknologi Sepuluh Nopember (ITS) Surabaya jurusan Teknik elektro dan mengambil konsentrasi dalam Bidang Studi Teknik Sistem Tenaga. Putra pertama dari dua bersaudara dari orang tua Bapak M. Hari Poernama dan Ibu Boedi Widyaningsih ini aktif dalam berbagai kegiatan, diantaranya HIMATEKTRO ITS, EEVENT ELEKTRO ITS. Penulis dapat dihubungi melalui alamat email : hardirizky@gmail.com